Recurrent Neural Network

CS109B Data Science 2

Pavlos Protopapas, Mark Glickman, and Chris Tanner

Outline

Motivation behind RNNs

Introduction to RNN

Training in RNN

Bidirectional RNNs

Deep RNNs

Flavors of RNN

Outline

Motivation behind RNNs

Introduction to RNN

Training in RNN

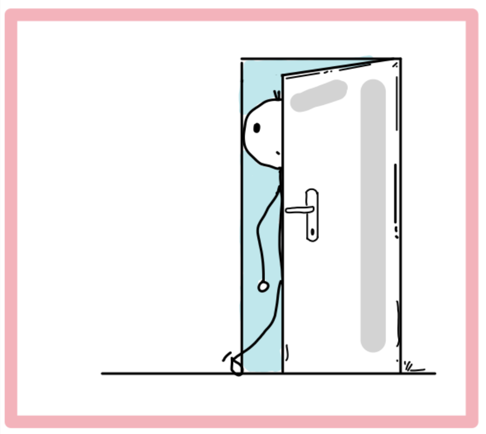
Bidirectional RNNs

Deep RNNs

Flavors of RNN

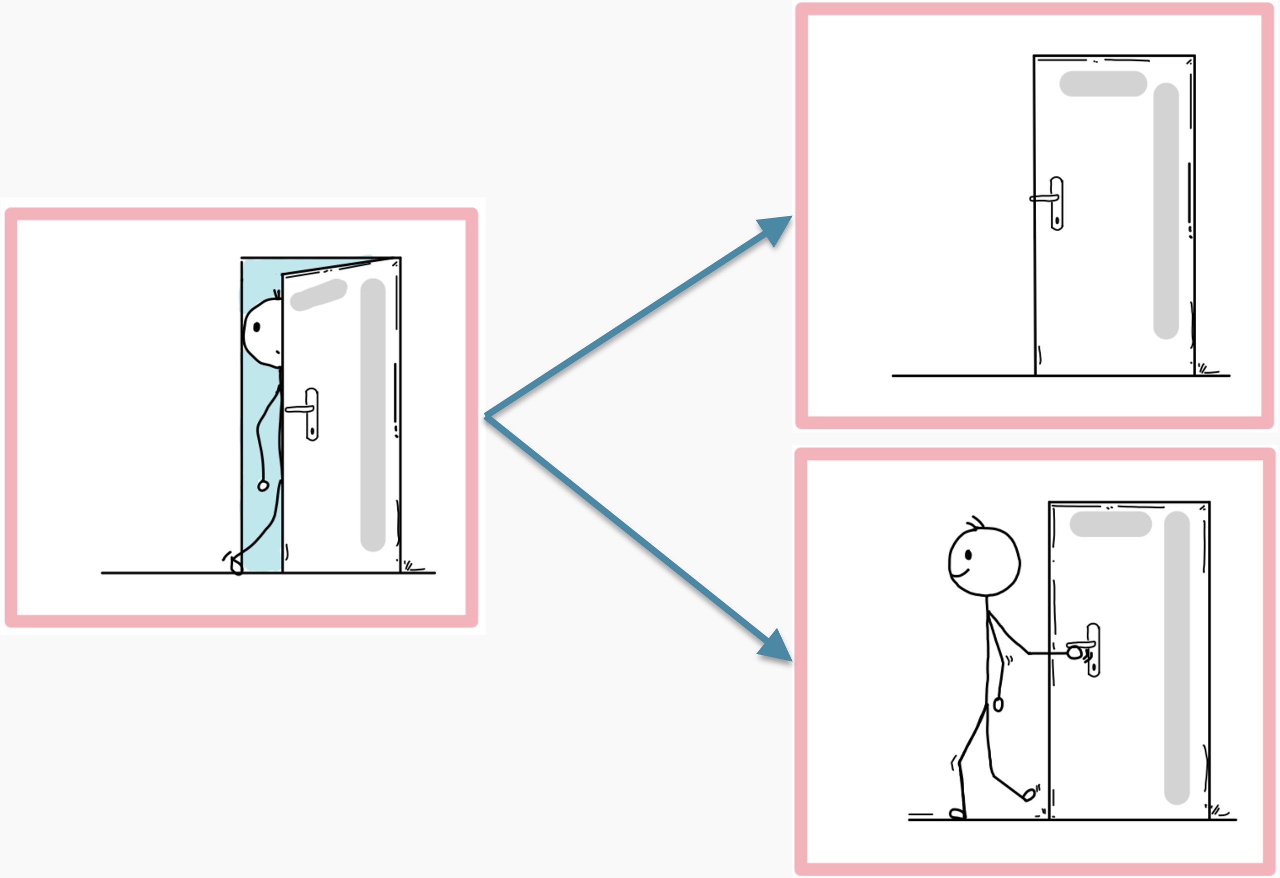
Motivation behind RNNs : **Sequences**

Given this frame, what do you think is the next likely frame?



Motivation behind RNNs : **Sequences**

Given this frame, what do you think is the next likely frame?

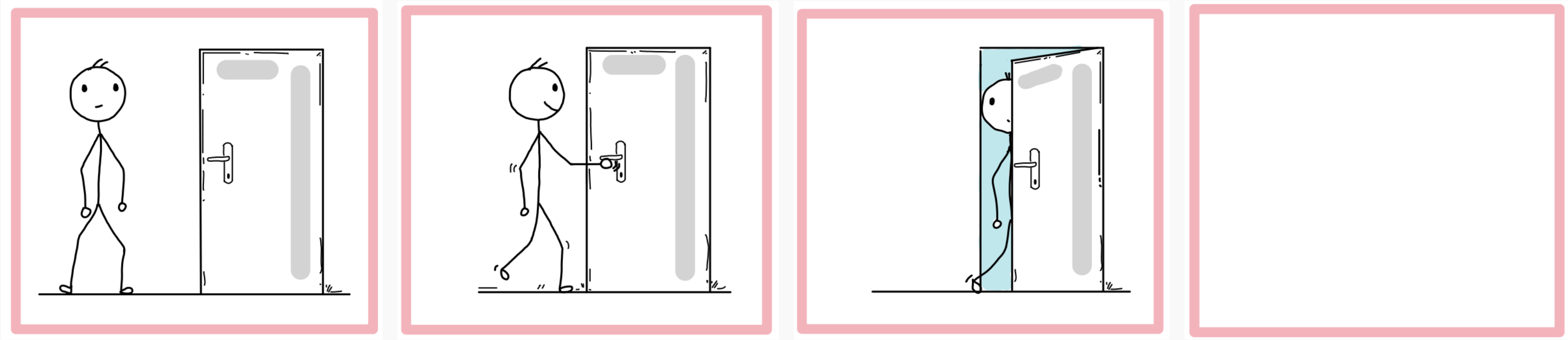


There are 2 options, either the person is going in or is coming out?

Based on only the central frame is would be difficult to predict the next frame.

Motivation behind RNNs : **Sequences**

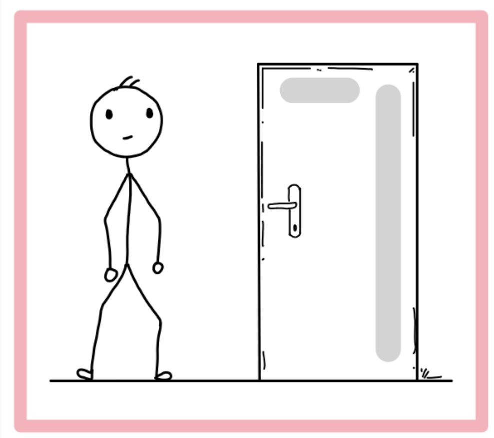
Frame 1 Frame 2 Frame 3 ?



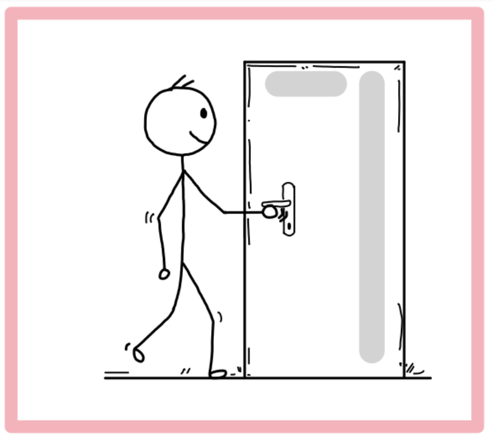
However, if we were to give the model the previous frames it would be easy to predict the next

Motivation behind RNNs : **Sequences**

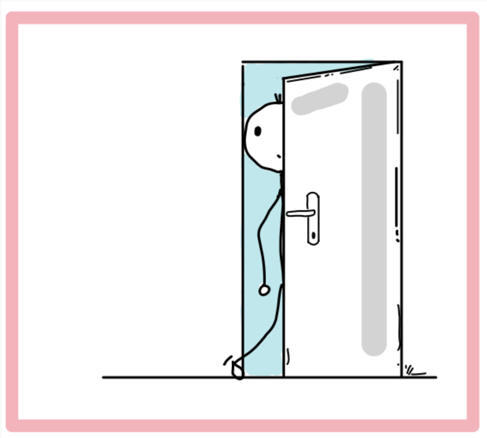
Frame 1



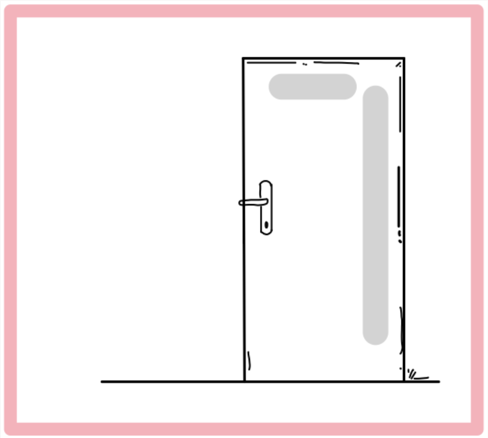
Frame 2



Frame 3



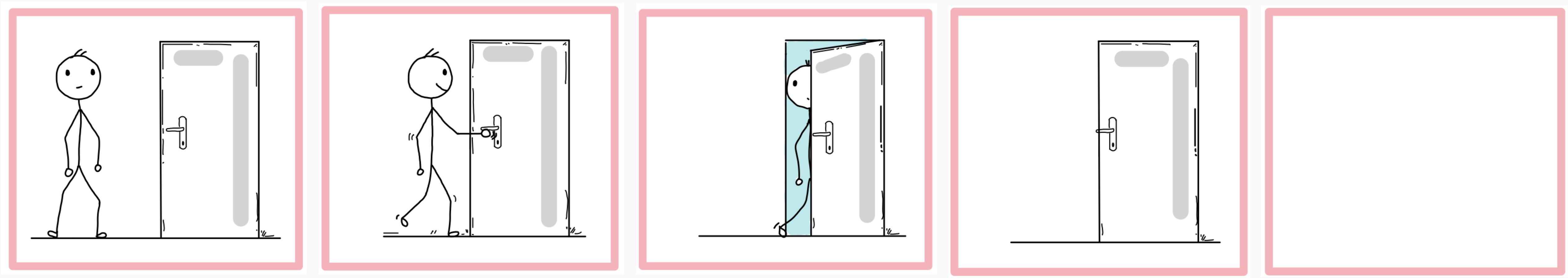
Frame 4



However, if we were to give the model the previous frames it would be easy to predict the next

Motivation behind RNNs : **Sequences**

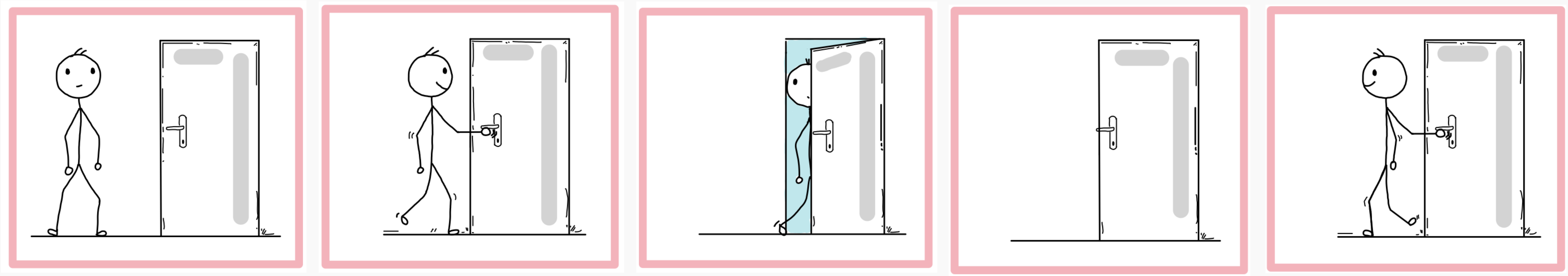
Frame 1 | Frame 2 | Frame 3 | Frame 4 | ?



However, if we were to give the model the previous frames it would be easy to predict the next

Motivation behind RNNs : **Sequences**

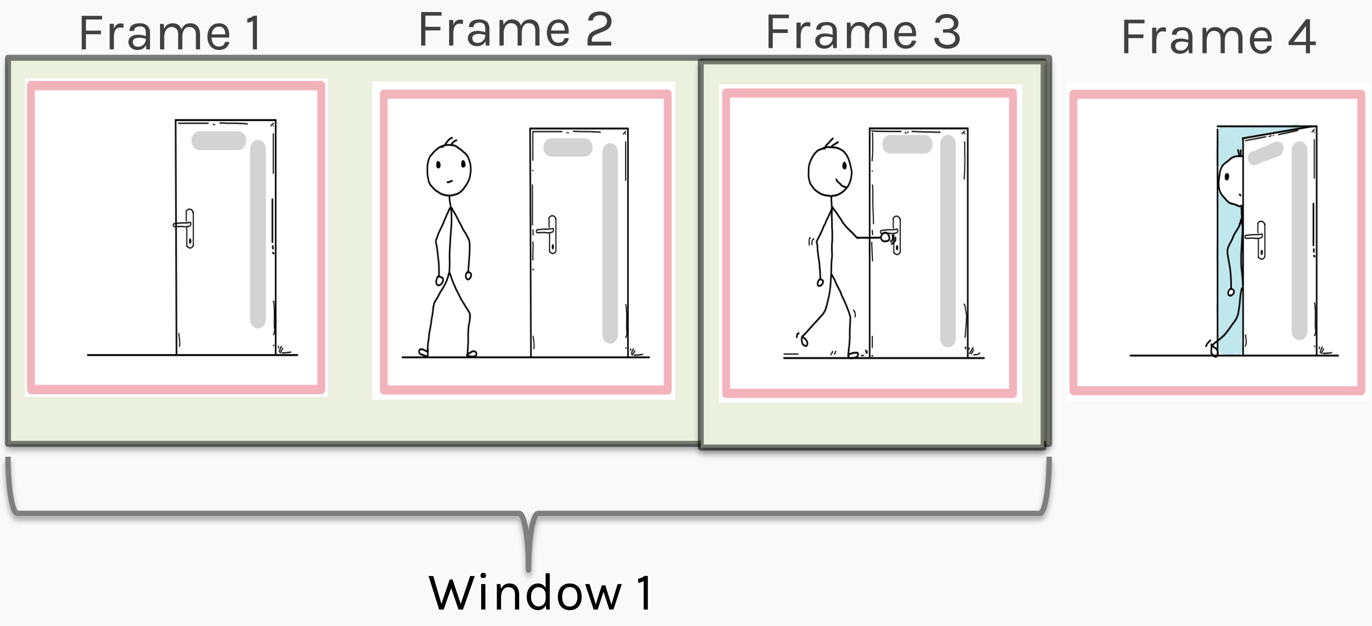
Frame 1 | Frame 2 | Frame 3 | Frame 4 | Frame 5



It would be easy for a model to predict the next frame if the previous sequence is provided.

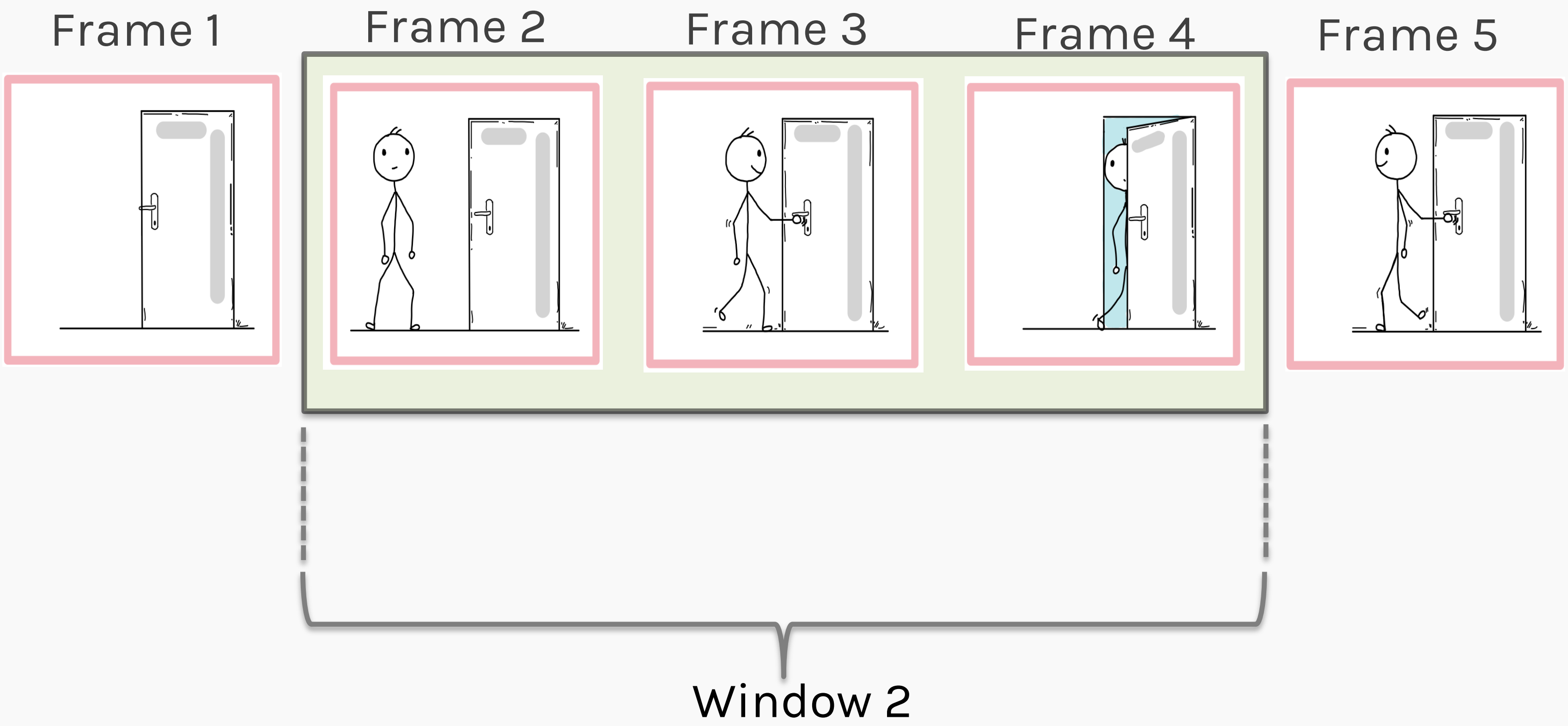
Sequences play an important role for forecasting and predictions.

Motivation behind RNNs : **Windowing**



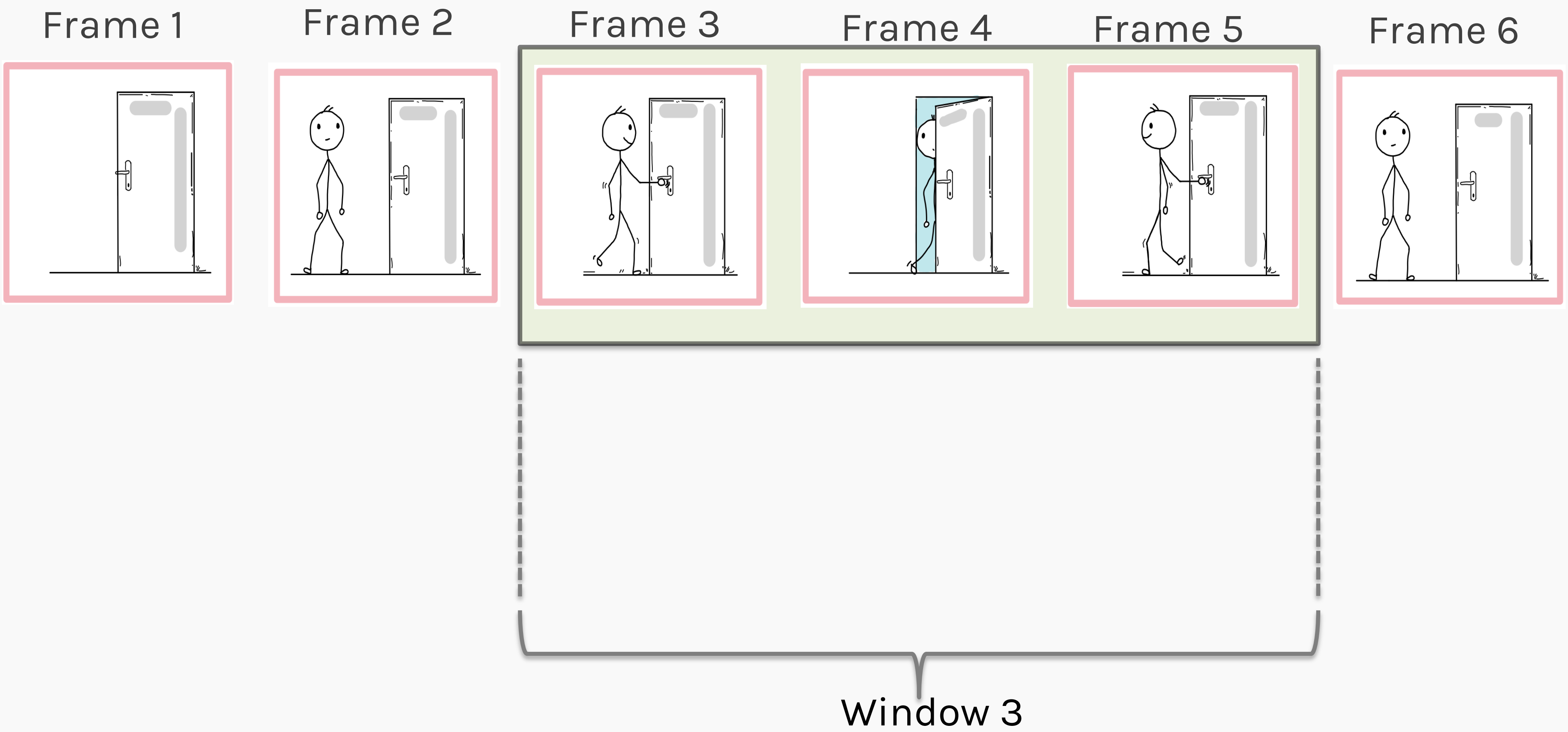
If we window a fixed number of frame (for example 3 here) as input to a neural network like FFNN or CNN then the prediction will work.

Motivation behind RNNs : **Windowing**



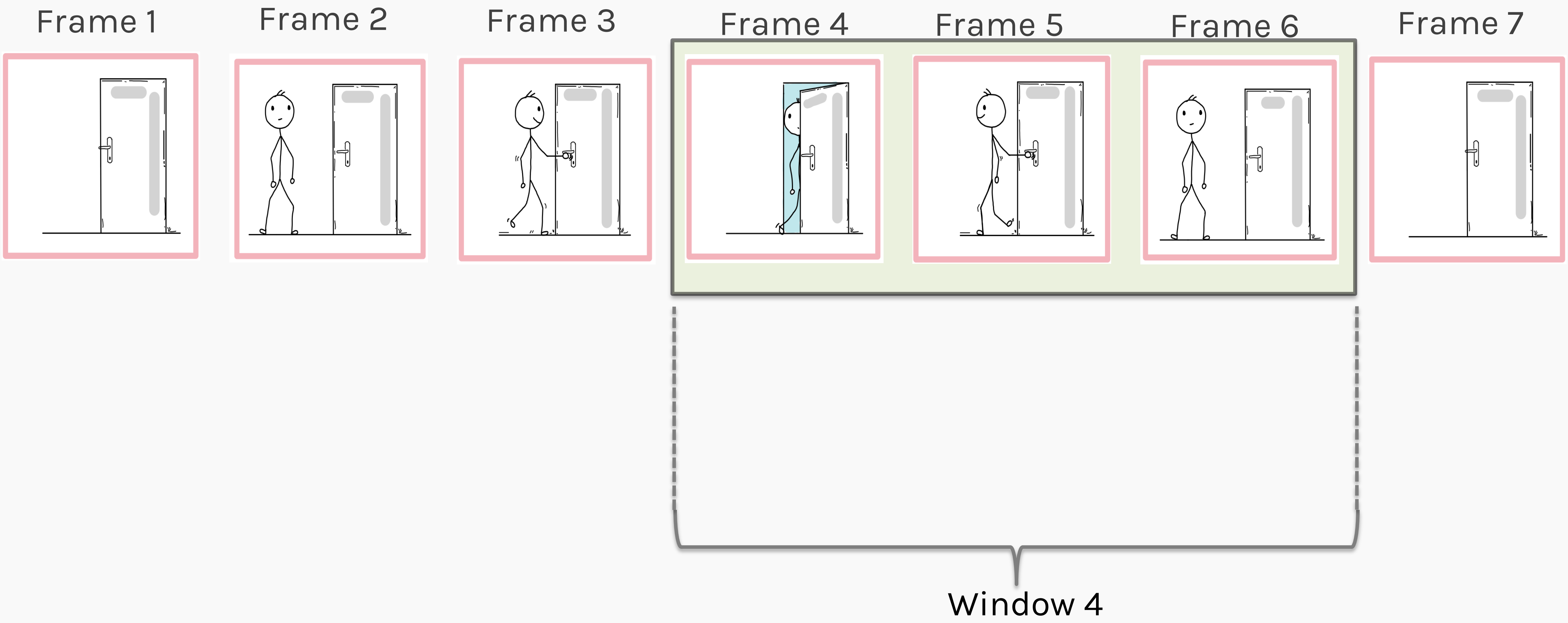
If we window a fixed number of frame (for example 3 here) as input to a neural network like FFNN or CNN then the prediction will work.

Motivation behind RNNs : **Windowing**



If we window a fixed number of frame (for example 3 here) as input to a neural network like FFNN or CNN then the prediction will work.

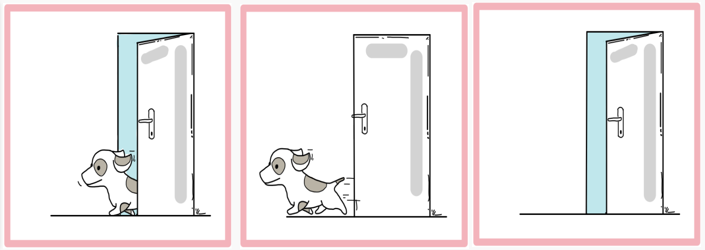
Motivation behind RNNs : **Windowing**



If we window a fixed number of frame (for example 3 here) as input to a neural network like FFNN or CNN then the prediction will work.

Motivation behind RNNs : **Windowing**

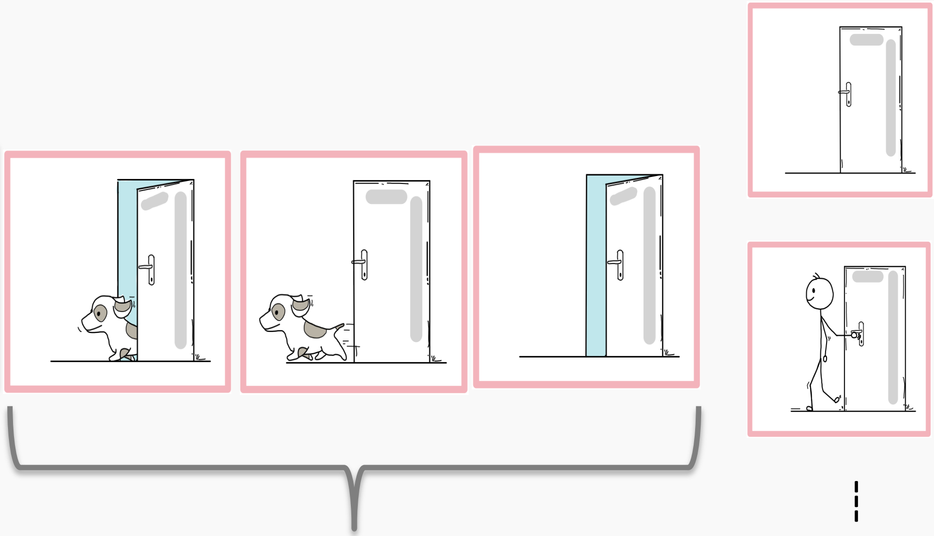
However, consider the following sequence of frames:



Window

Motivation behind RNNs : **Windowing**

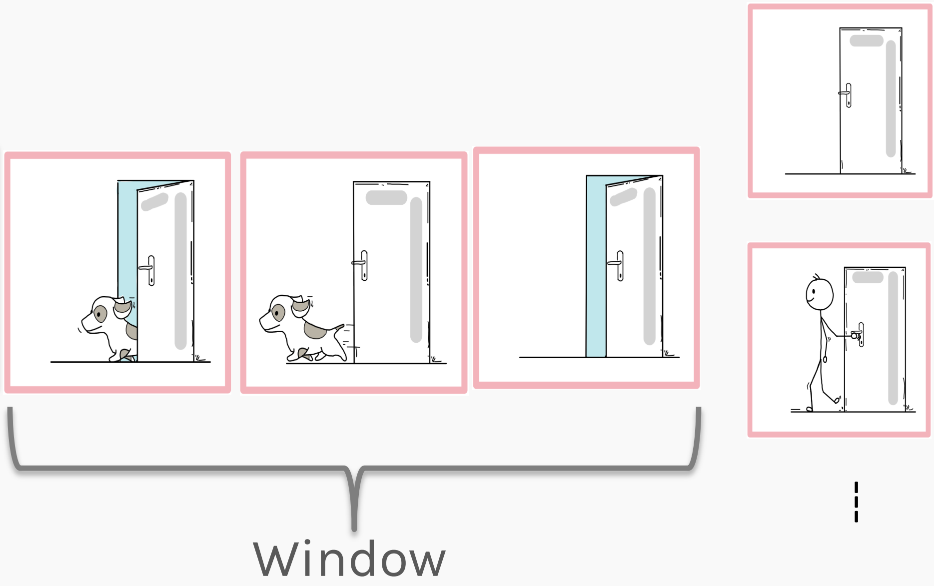
However, consider the following sequence of frames:



There are many options, either the person coming out, or the door shut, etc.

Motivation behind RNNs : **Windowing**

However, consider the following sequence of frames:

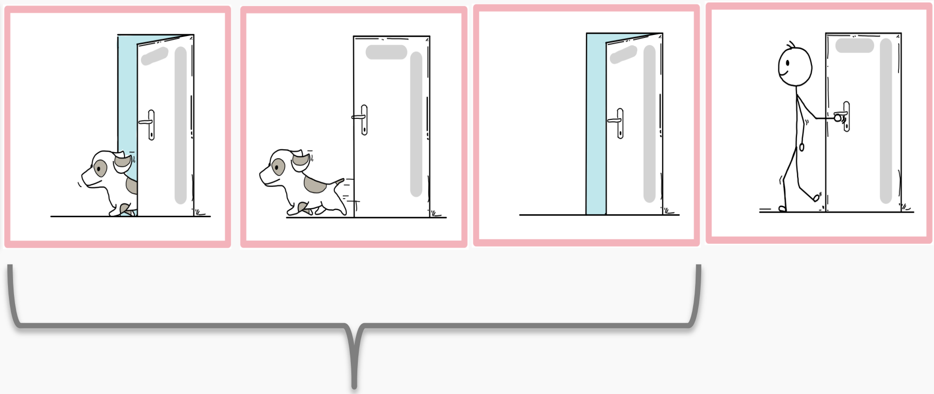


There are many options, either the person coming out, or the door shut, etc.

Thus, a longer memory is needed.

Motivation behind RNNs : **Windowing**

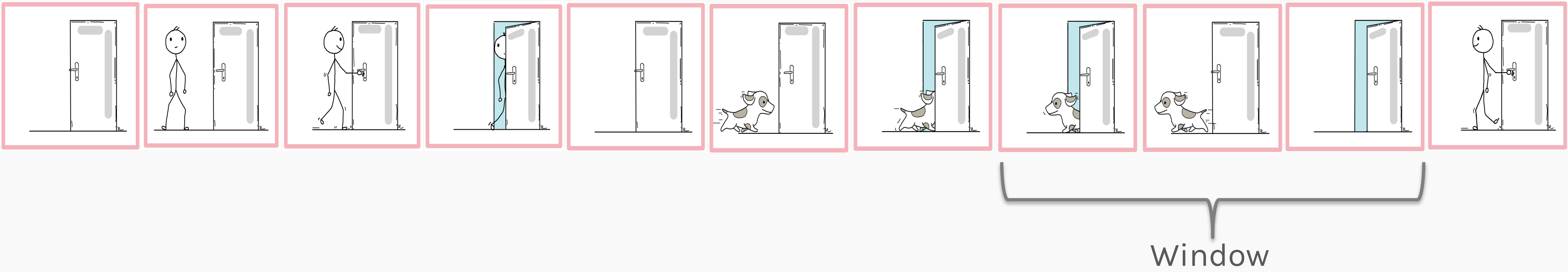
However, consider the following sequence of frames:



To predict the next frame, the window size is not enough.
Thus, a longer memory is needed.

Motivation behind RNNs : **Windowing**

However, consider the following sequence of frames:



To predict the next frame, the window size is not enough.

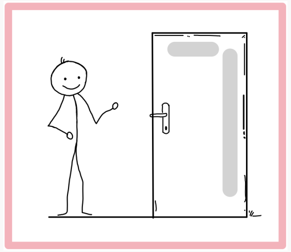
Thus, a longer memory is needed.

RNNs introduce a concept of memory and carry forward the context history.

Motivation behind RNNs : **Ordering**

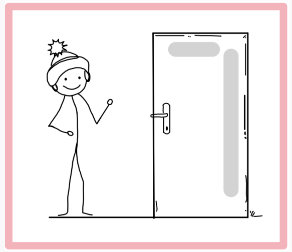
Consider the following sequence of frames given as input to a FFNN:

Frame 1



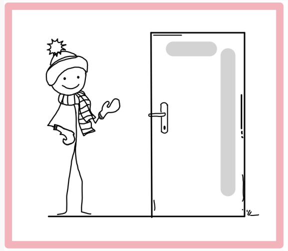
(wakeup)

Frame 2



(wear hat)

Frame 3

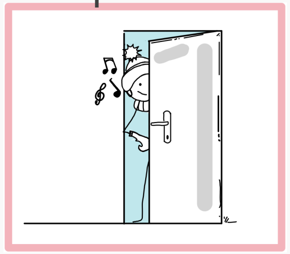


(wear gloves)

Frame 4

(listen to music)

To predict

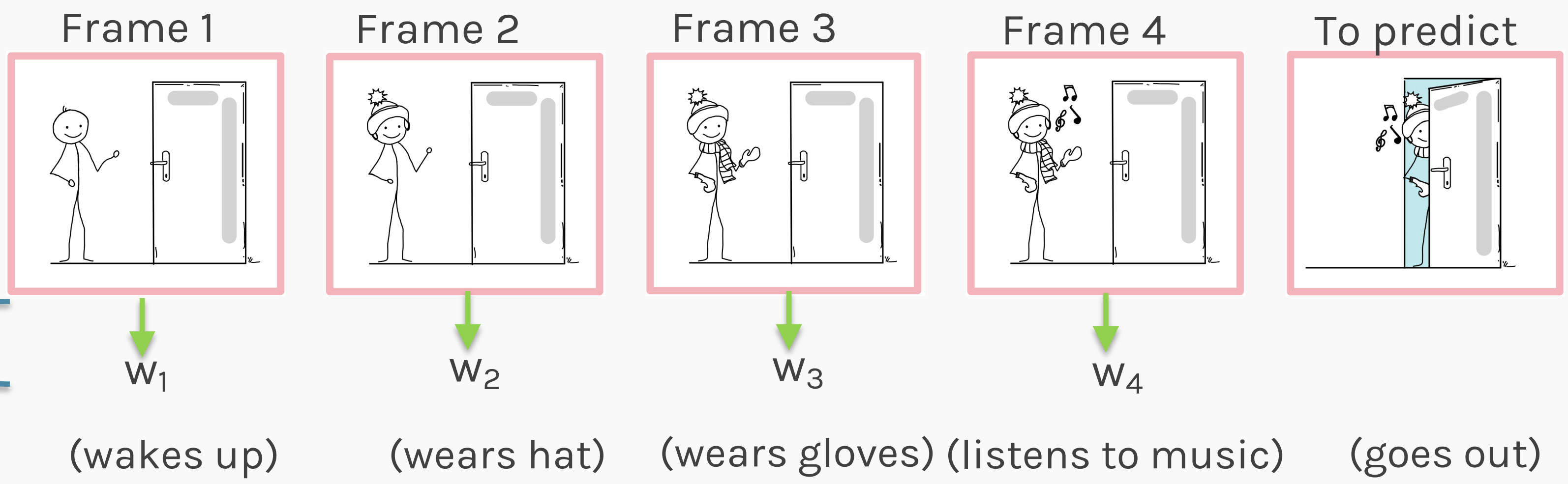


(go out)

Motivation behind RNNs : **Ordering**

Consider the following sequence of frames given as input to a FFNN:

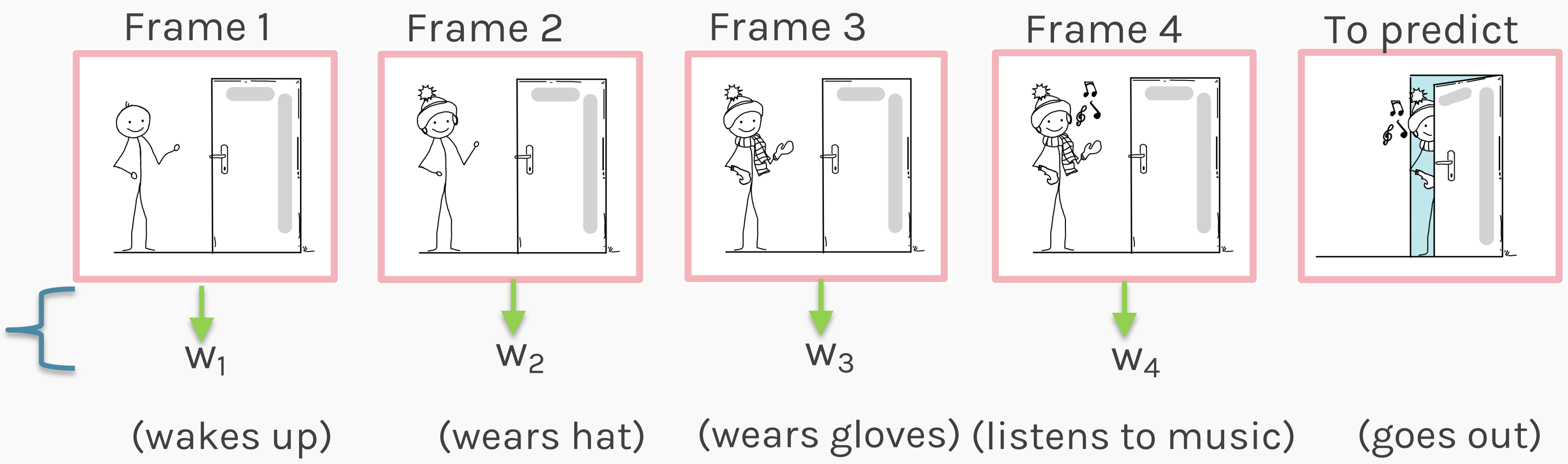
After training each frame has a weight associated to it.



Motivation behind RNNs : **Ordering**

Consider the following sequence of frames given as input to a FFNN:

After training each frame has a weight associated to it.



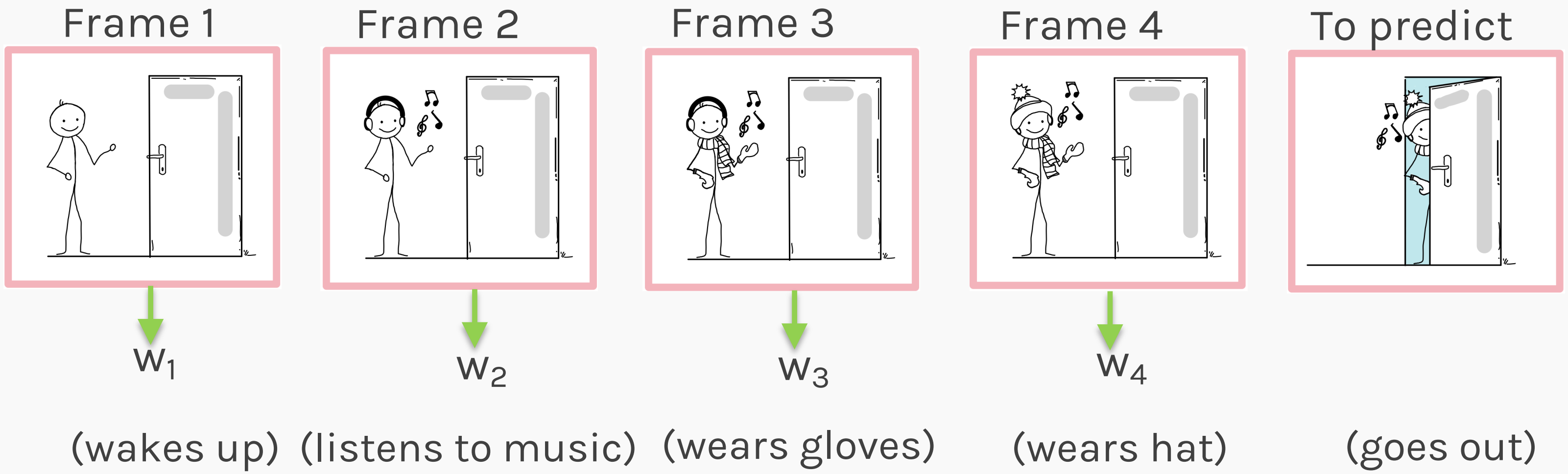
Here w_4 (listen to music) has the lowest weight ~ 0 i.e., it is the least important to predict the next frame as it has nothing to do with going out.

w_2 (wear hat) has the highest value i.e., it is the most important to predict the next frame, followed by w_3 (wear gloves).

Motivation behind RNNs : **Ordering**

The **order** of the frames is now slightly changed and given as input to the FFNN:

The same trained weights are used regardless of the order.



Though this order also makes sense for predicting going out, the prediction will be different this time as wear hat which was most responsible for prediction is transformed using w_4, which is ~ 0. The same is the case for wear gloves.

However, listening to music that is not relevant for prediction is highly important by transforming using w_2.

Motivation behind RNNs : **Summary**

RNNs exhibit the following advantages for sequence modelling:

- Handle variable-length sequences
- Keep track of long-term dependencies
- Maintain information about the order as opposed to FFNN
- Share parameters across the network

Outline

Motivation behind RNNs

Introduction to RNN

Training in RNN

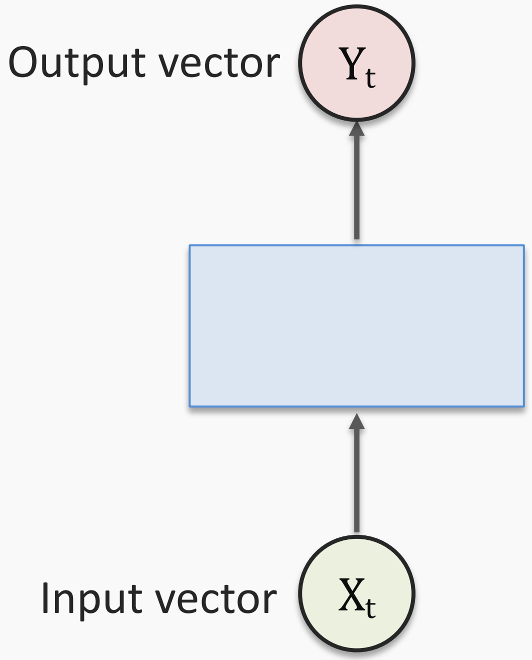
Bidirectional RNNs

Deep RNNs

Flavors of RNN

Introduction to RNNs

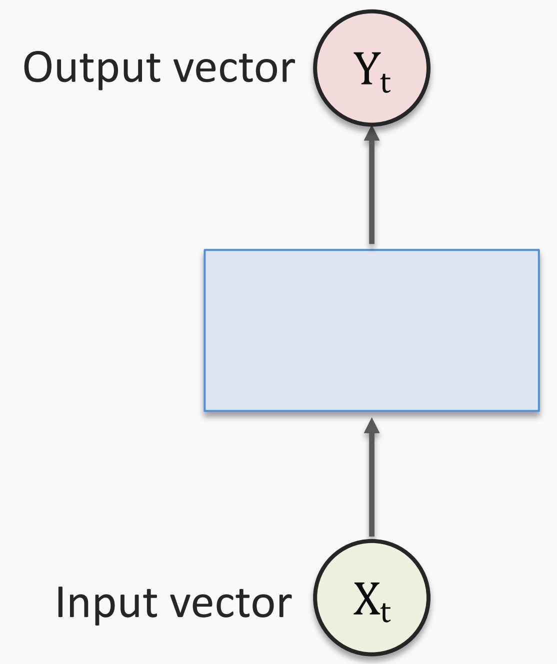
FEED FORWARD
NEURAL NETWORK



- Cannot maintain previous information

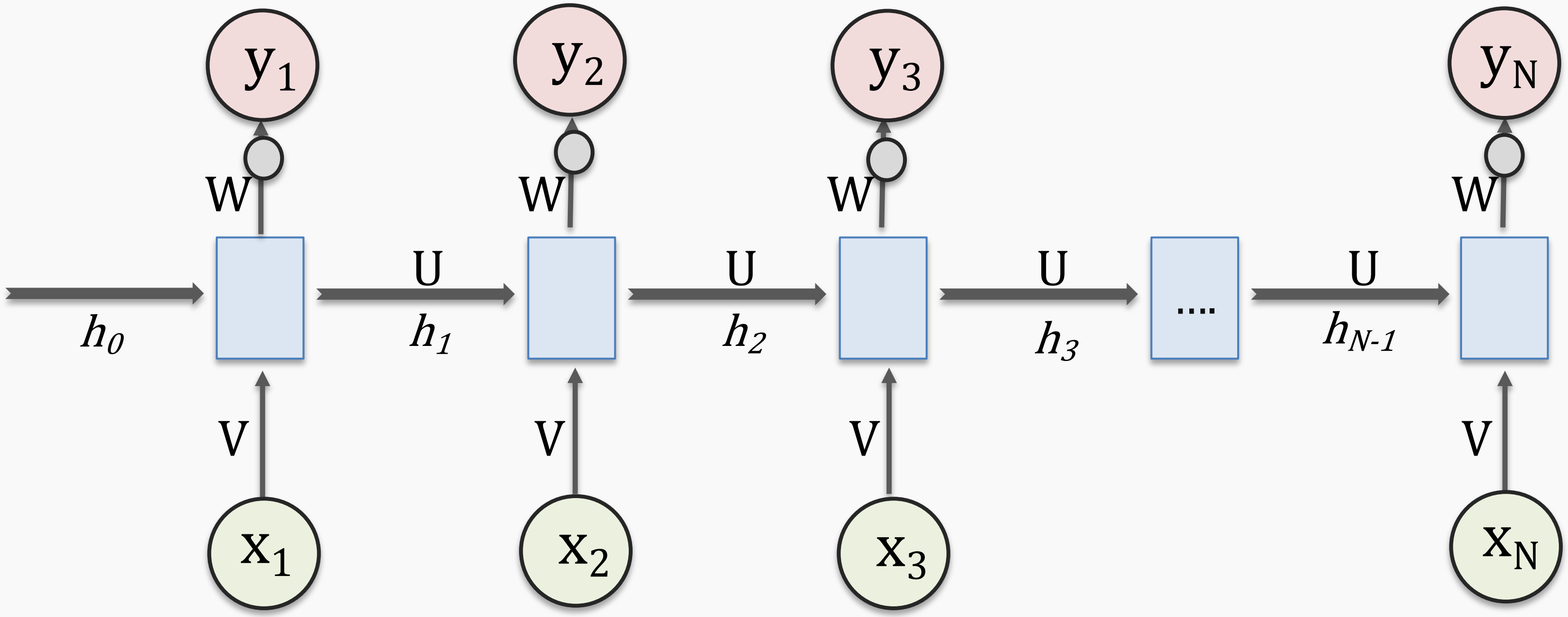
Introduction to RNNs

FEED FORWARD NEURAL NETWORK



- Cannot maintain previous information

RECURRENT NEURAL NETWORK

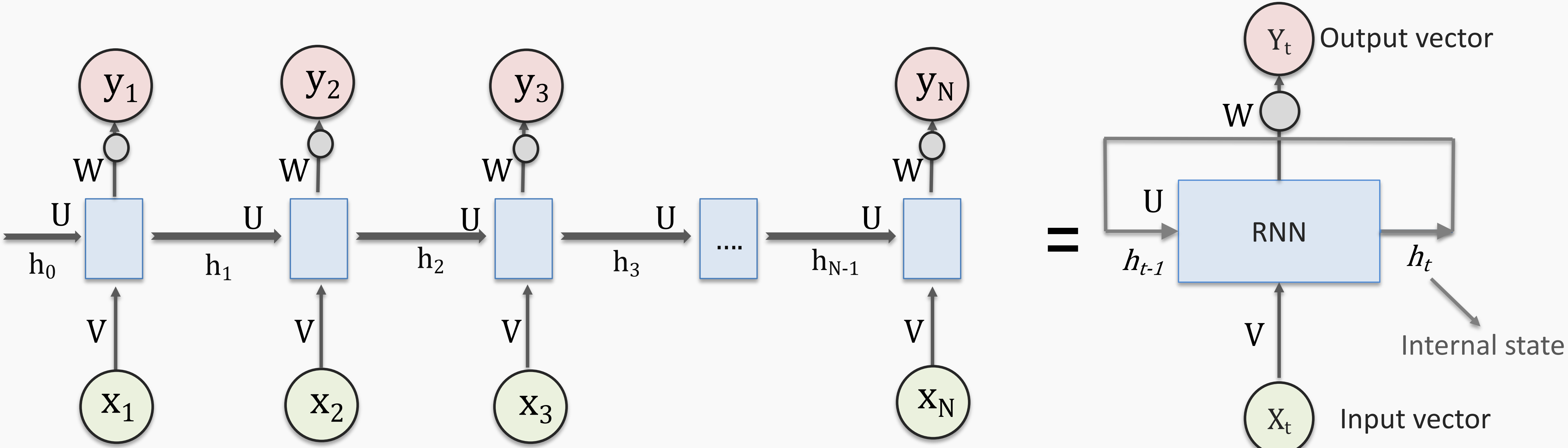


The term **recurrent** comes from the fact that information is being passed from one time step to the next internally within the network.

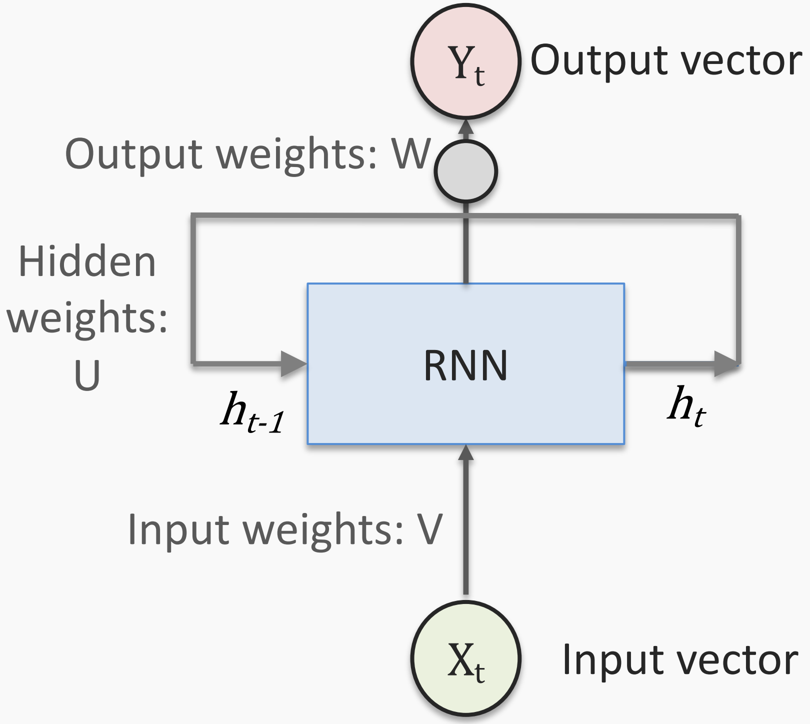
Network has loops for information to persist over time

Introduction to RNNs

Alternative short representation:



Introduction to RNNs

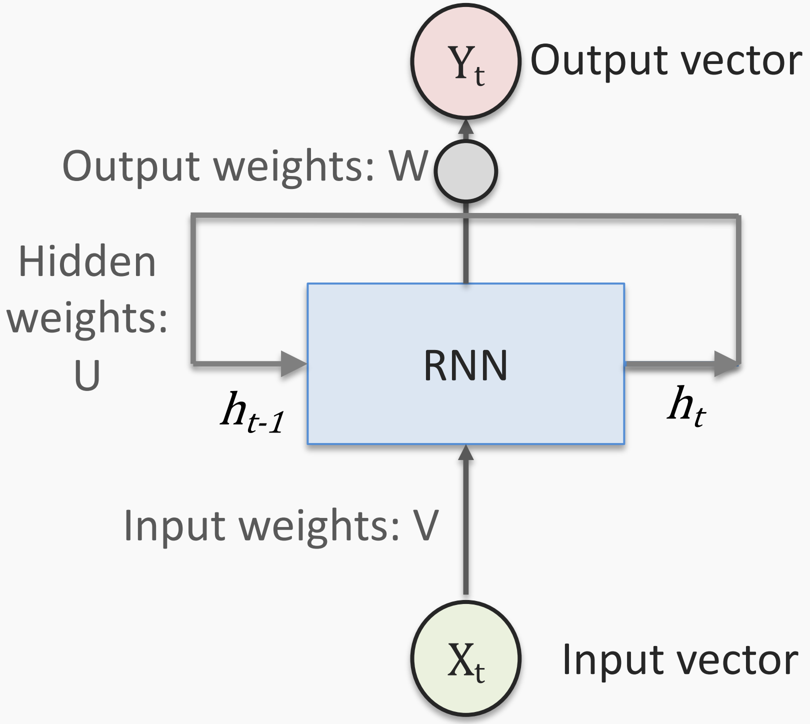


RNNs are governed by a **recurrence relation** applied at every time step for a given sequence.

$$h_t = f_{u,v}\,(\,h_{t-1}\,,\,x_t\,)$$

At each time step the RNN is fed the current input and the previous hidden state.

Introduction to RNNs



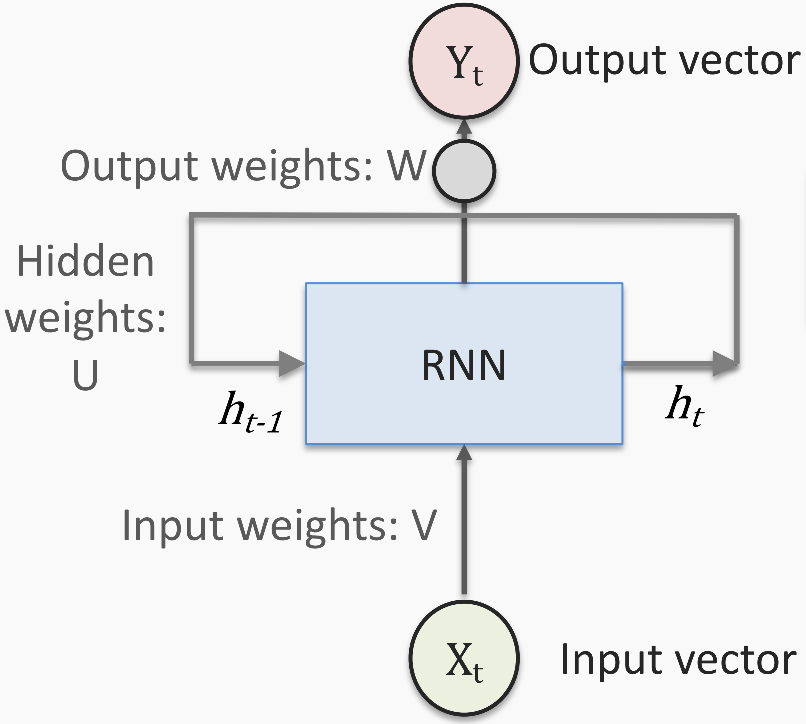
RNNs are governed by a **recurrence relation** applied at every time step for a given sequence.

$$\boxed{h_t} = \boxed{f_{u,v}}(\boxed{h_{t-1}}, \boxed{x_t})$$

State — Function parameterized by u,v — Old State — Input vector at time step t

The function f_w and the parameters used for all time steps are learned during training.

Introduction to RNNs



RNNs are governed by a **recurrence relation** applied at every time step for a given sequence.

Multiple names:
- Hidden state
- State
- Encoding
- Embedding

$$\boxed{h_t} = \boxed{f_{u,v}}\left(\boxed{h_{t-1}}, \boxed{x_t}\right)$$

Function parameterized by u,v

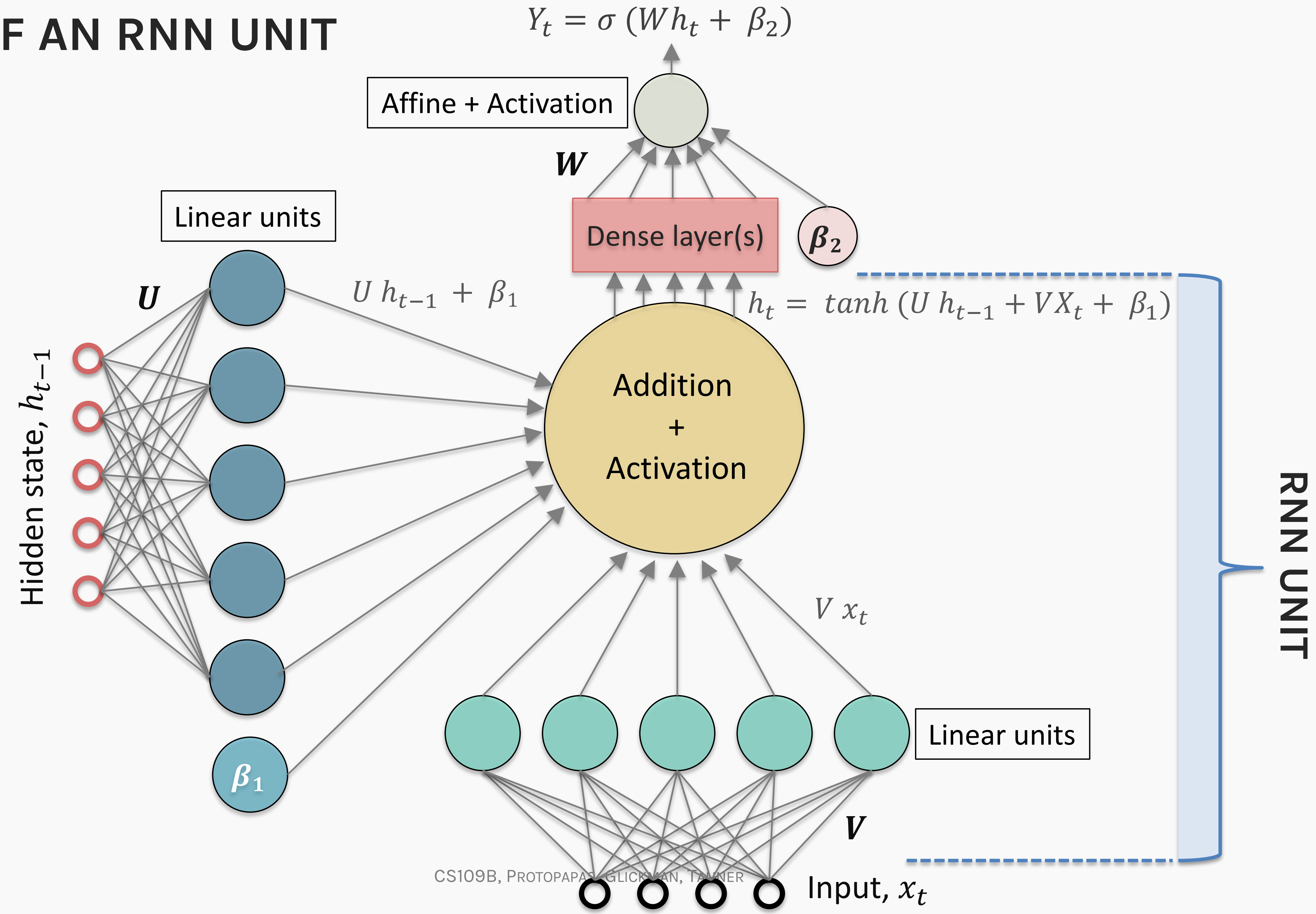
Old State

Input vector at time step t

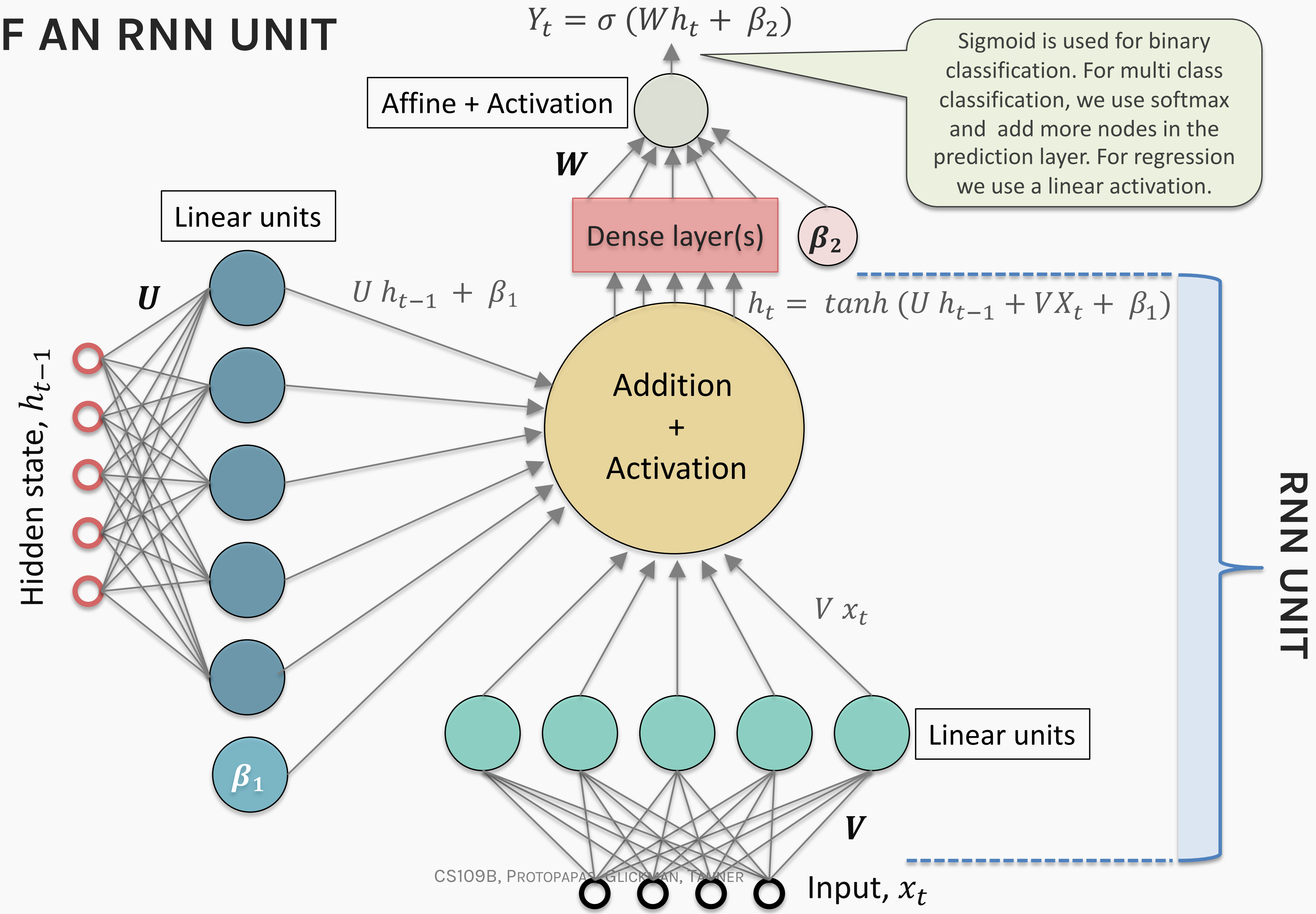
We often ignore to mention the bias here. It should be: $f_{u,v,\beta}()$

The function f_W and the parameters used for all time steps are learned during training.

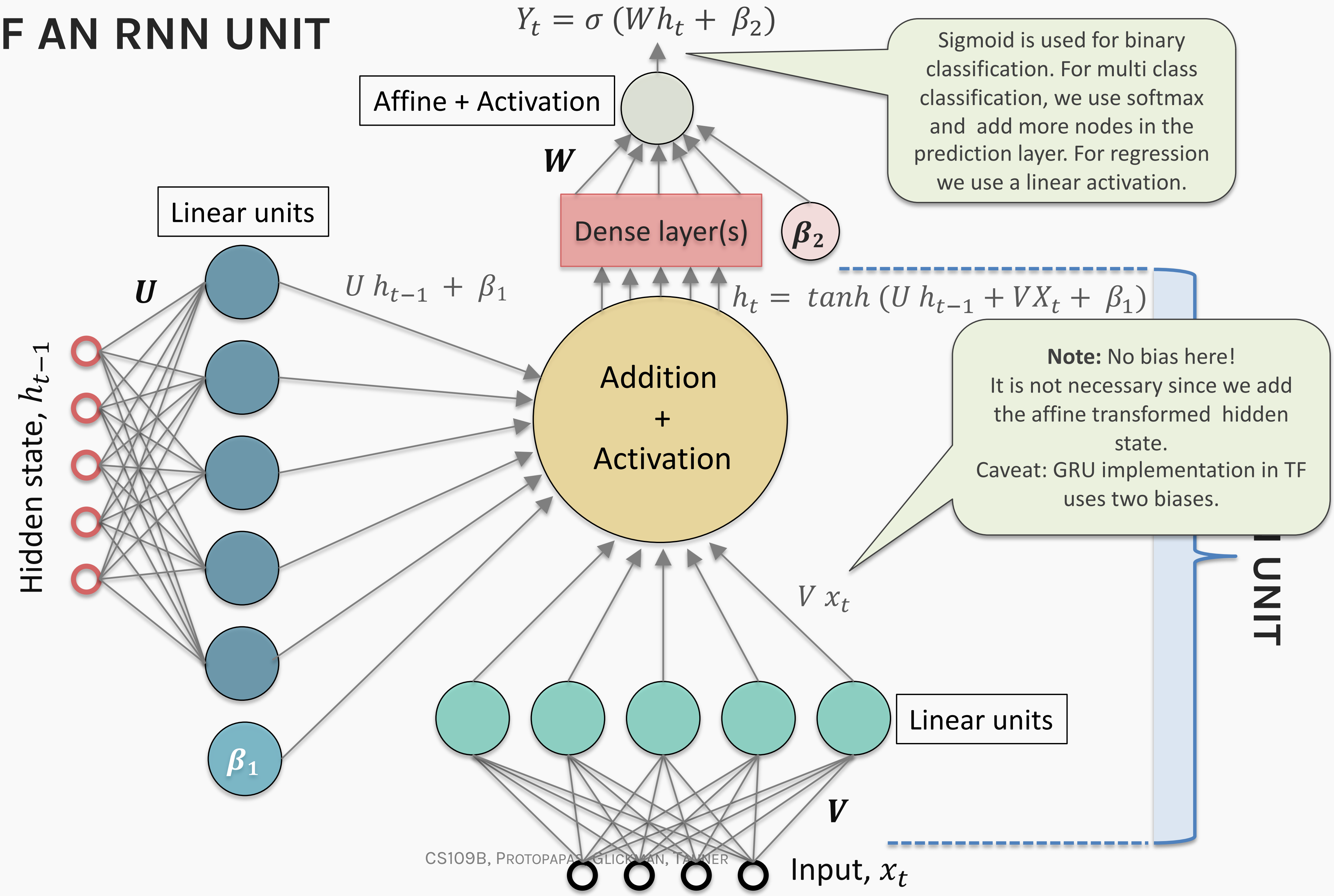
ANATOMY OF AN RNN UNIT



ANATOMY OF AN RNN UNIT



ANATOMY OF AN RNN UNIT




CS109B
Endgame

Outline

Motivation behind RNNs

Introduction to RNN

Training in RNN

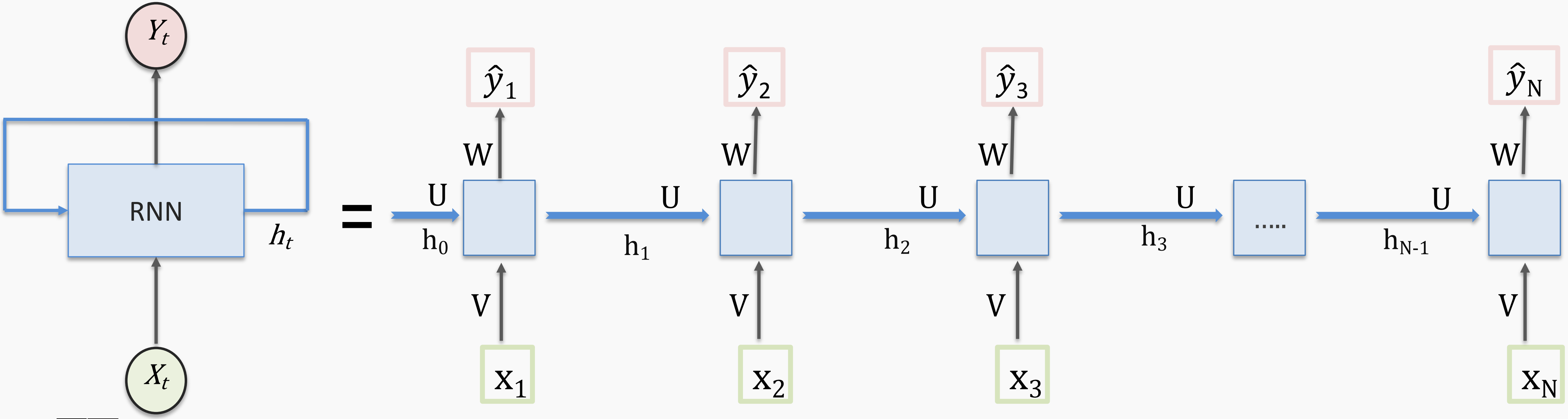
Bidirectional RNNs

Deep RNNs

Flavors of RNN

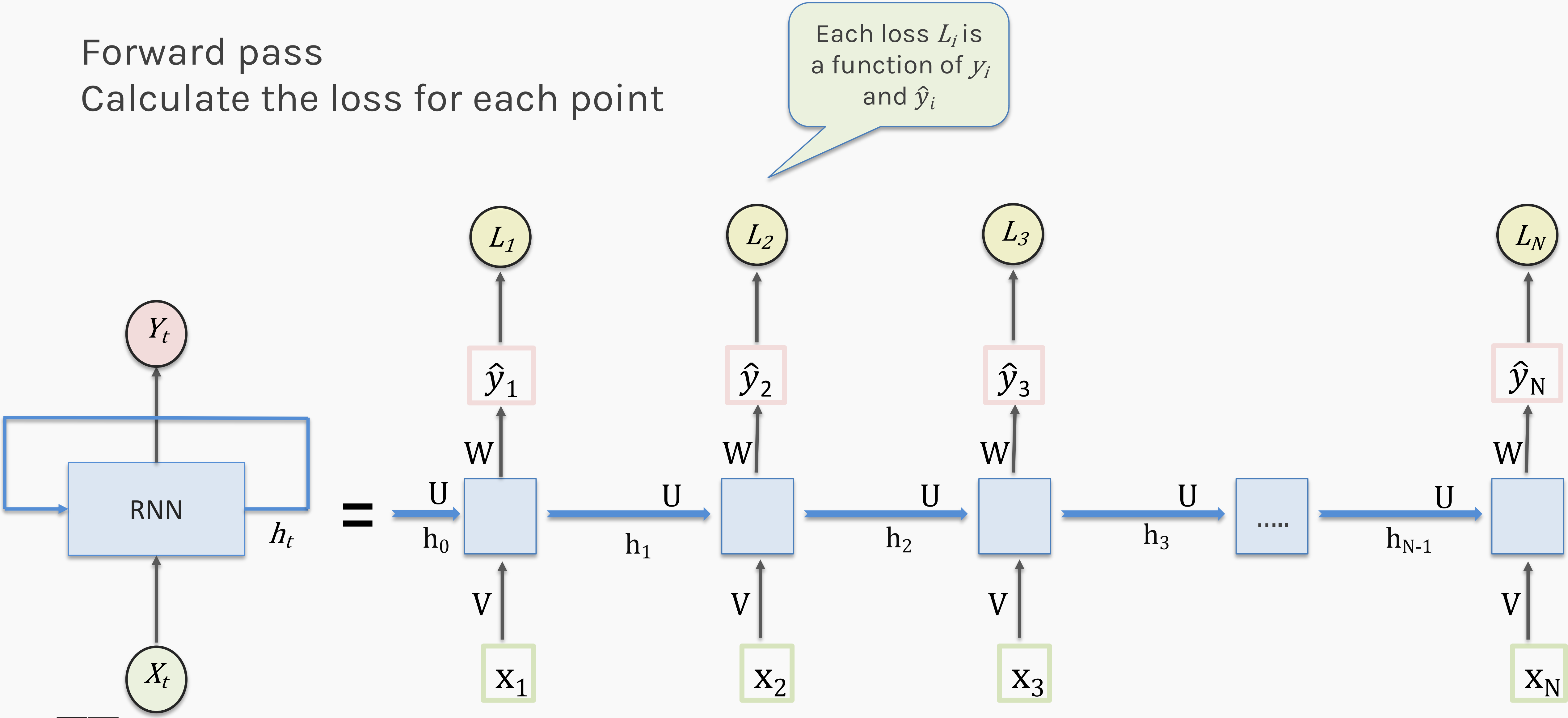
Training in RNN

Forward pass



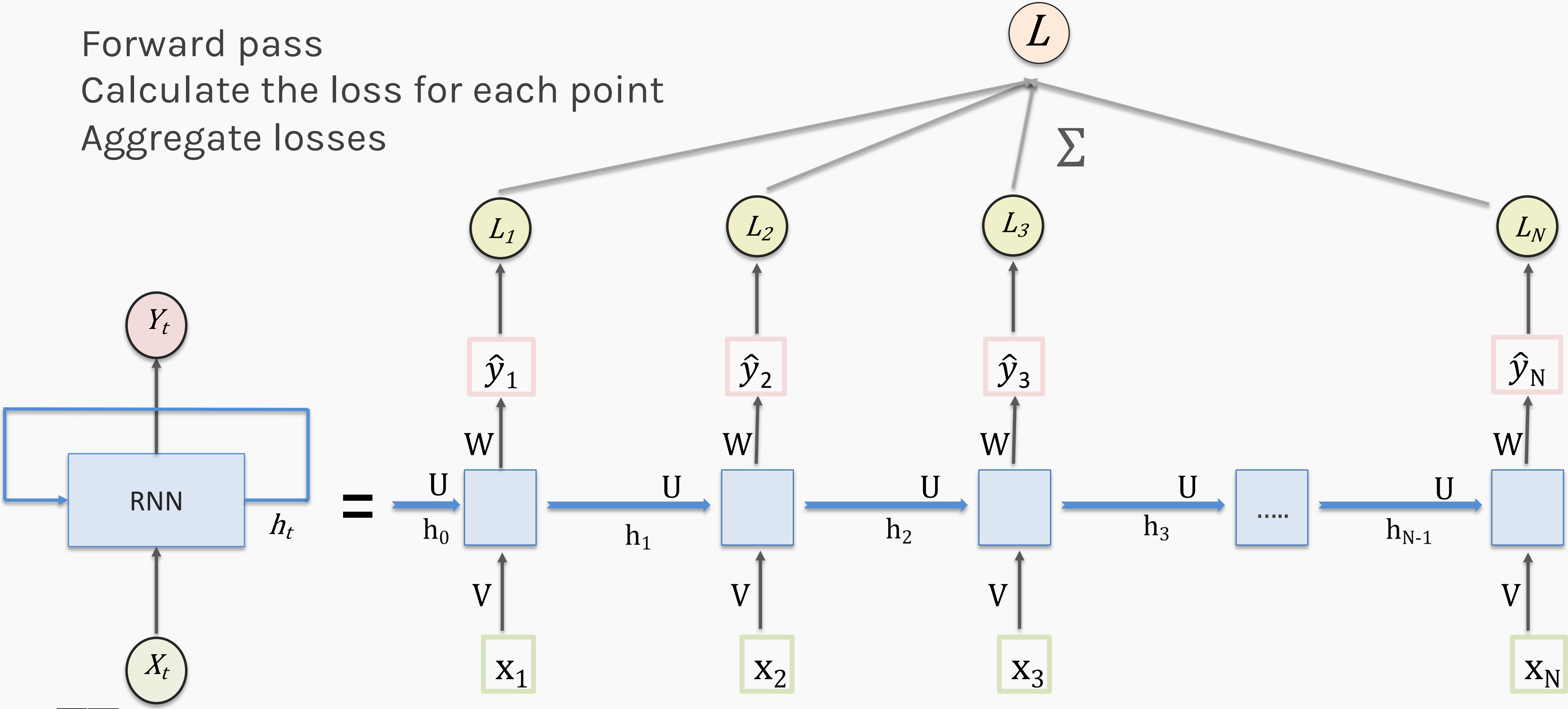
Training in RNN

Forward pass
Calculate the loss for each point



Training in RNN

Forward pass
Calculate the loss for each point
Aggregate losses

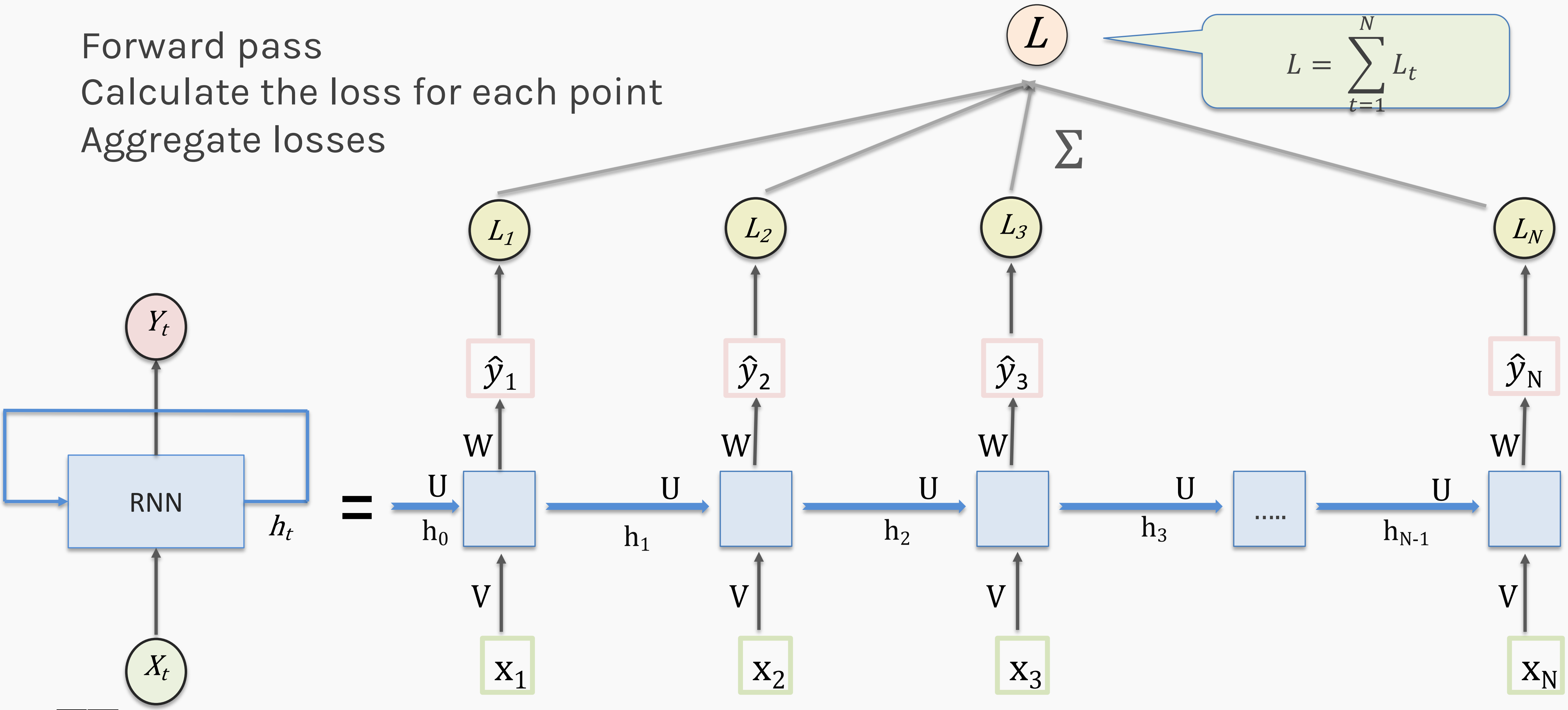


Training in RNN

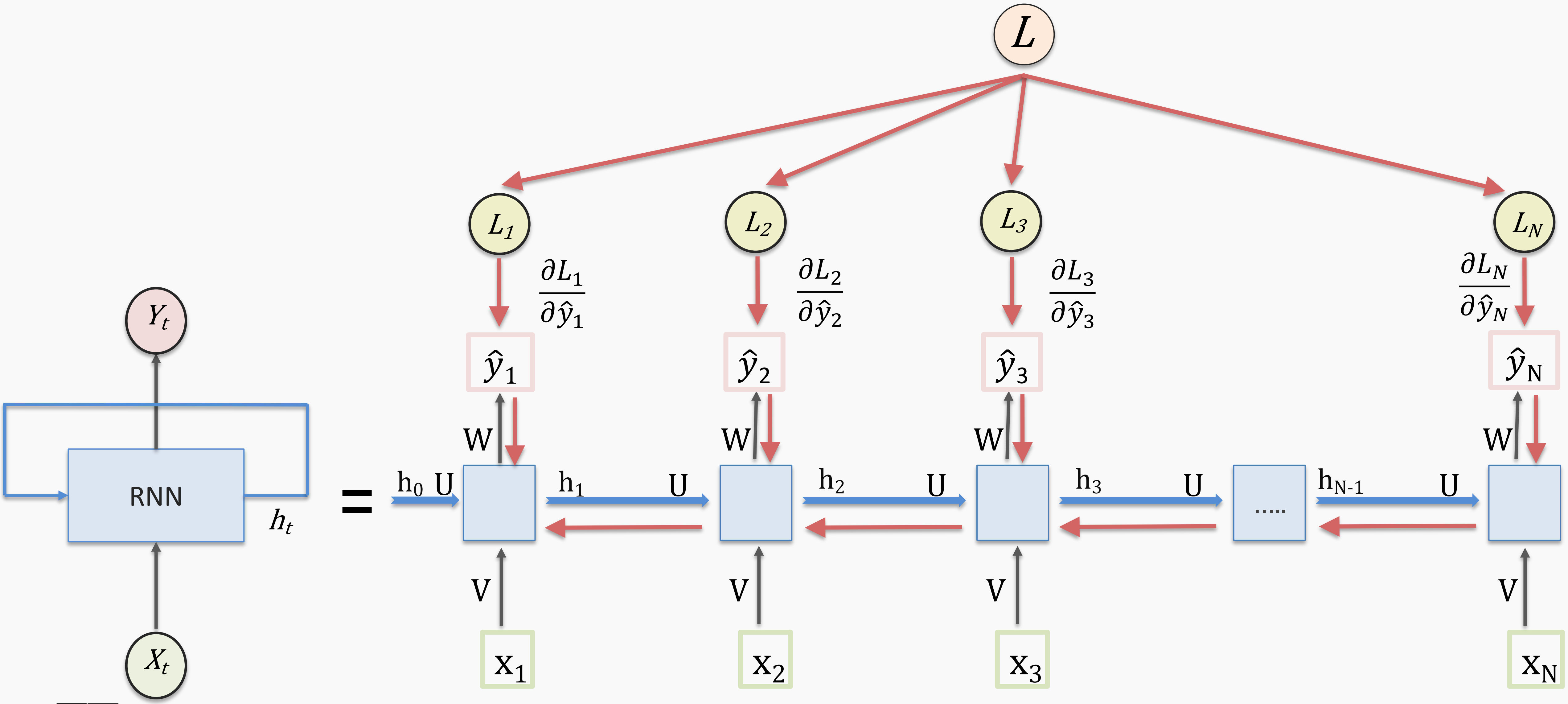
Forward pass

Calculate the loss for each point

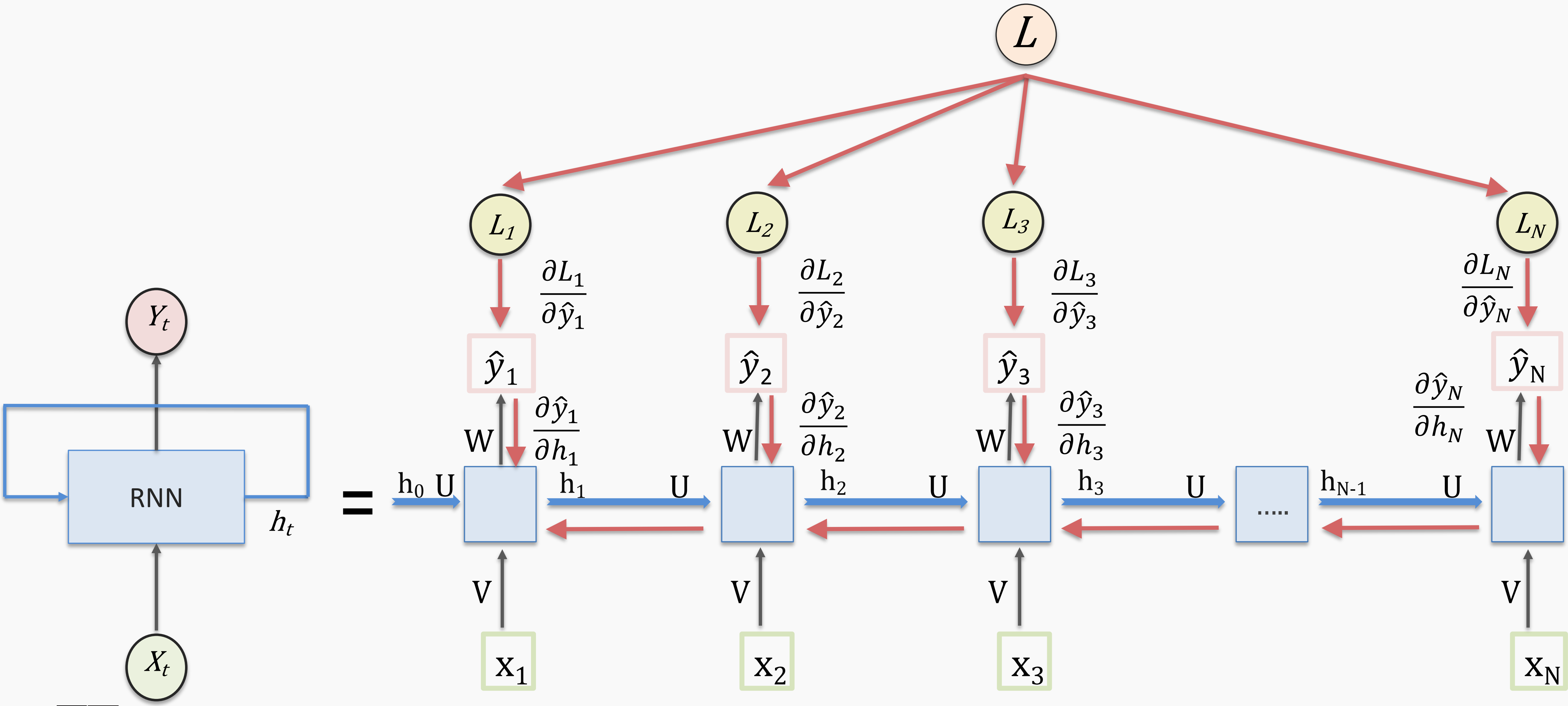
Aggregate losses



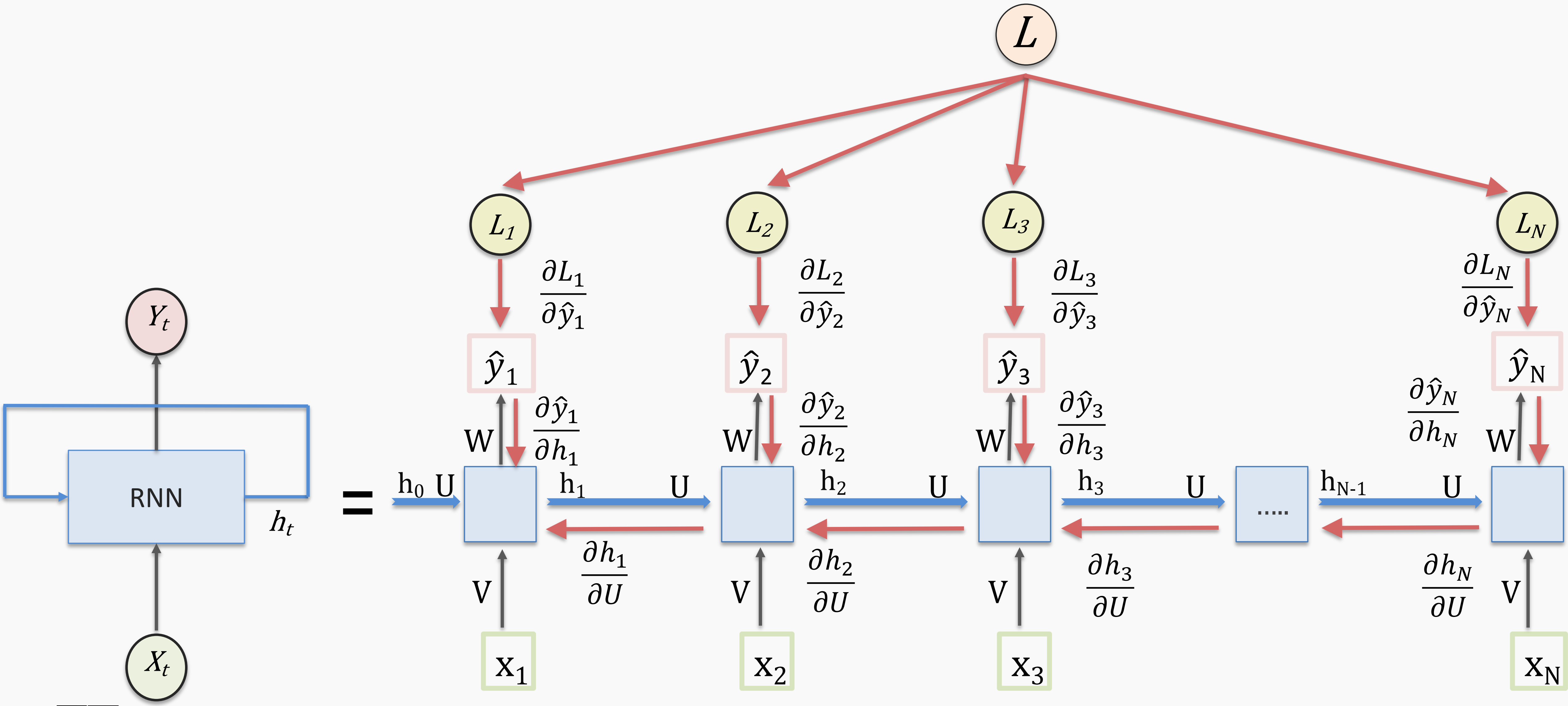
Training in RNN: **Backpropagation**



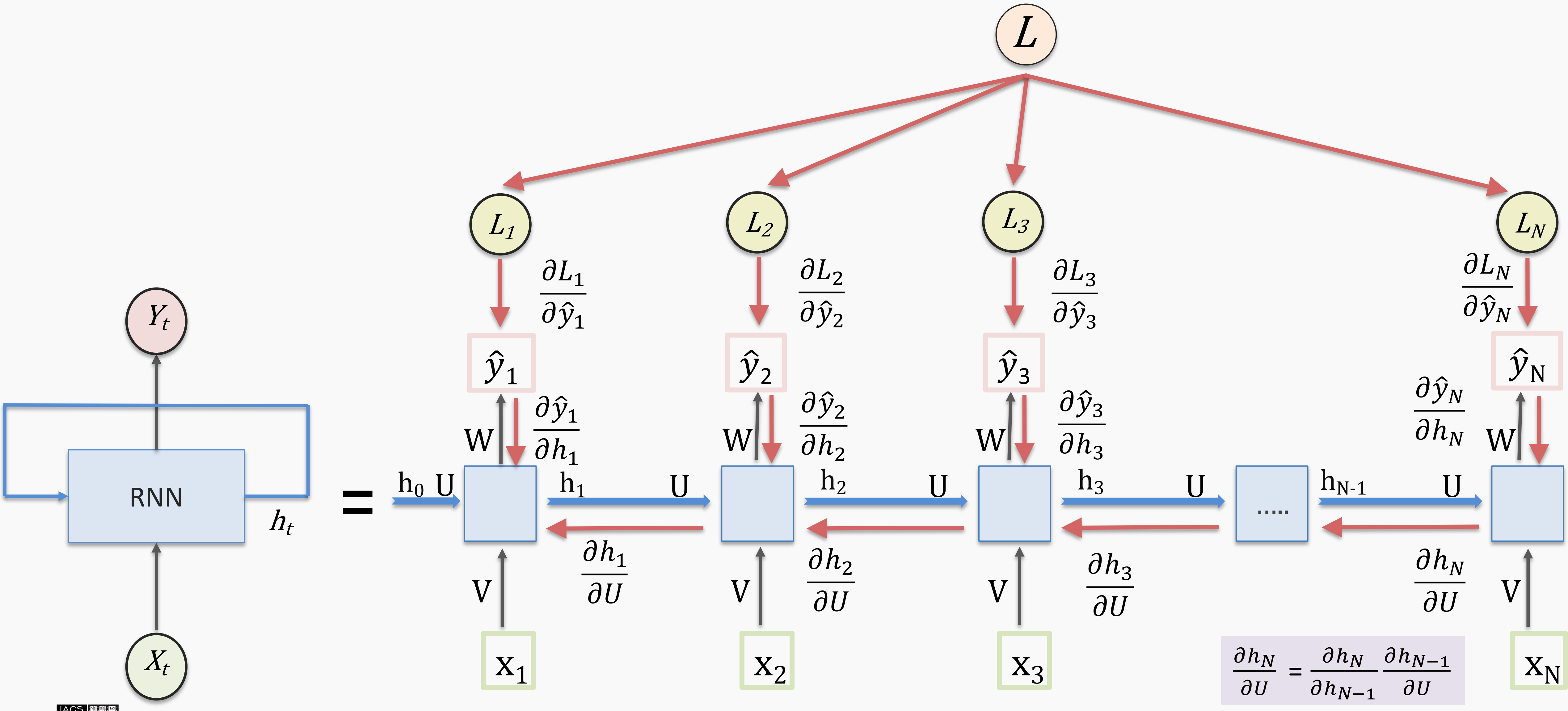
Training in RNN: **Backpropagation**



Training in RNN: **Backpropagation**



Training in RNN: Backpropagation



Training in RNN: **Backpropagation**

During backpropagation for each parameter at each time step *i*, a gradient is computed.

The individual gradients computed are then averaged at time step *t* and used to update the entire network.

The error flows back in time.

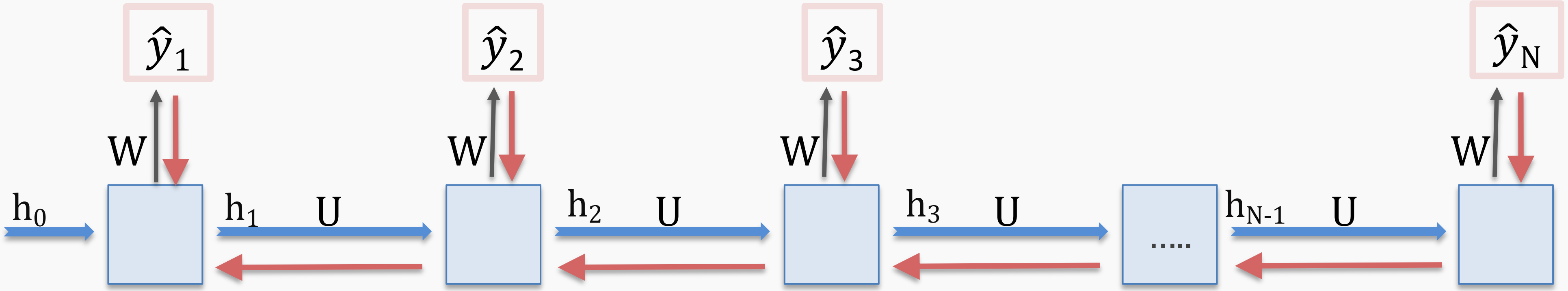
$$\frac{dL}{dU} = \sum_t \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial U}$$

$$\frac{\partial h_t}{\partial U} = \sum_{k=1}^{t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial U}$$

$$\frac{\partial h_t}{\partial h_k} = \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial h_{t-2}} \cdots \frac{\partial h_{k+1}}{\partial h_k} = \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}}$$

$$\frac{\partial L_t}{\partial U} = \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \left(\frac{dh_t}{dU} + \frac{dh_t}{dh_{t-1}} \frac{dh_{t-1}}{dU} + \frac{dh_t}{dh_{t-1}} \frac{dh_{t-1}}{dh_{t-2}} \frac{dh_{t-2}}{dU} + \cdots \right)$$

Training in RNN: **Backpropagation issues**



For longer sentences, we must backpropagate through more time steps.

This requires the gradient to be multiplied many times which cause the following issues.

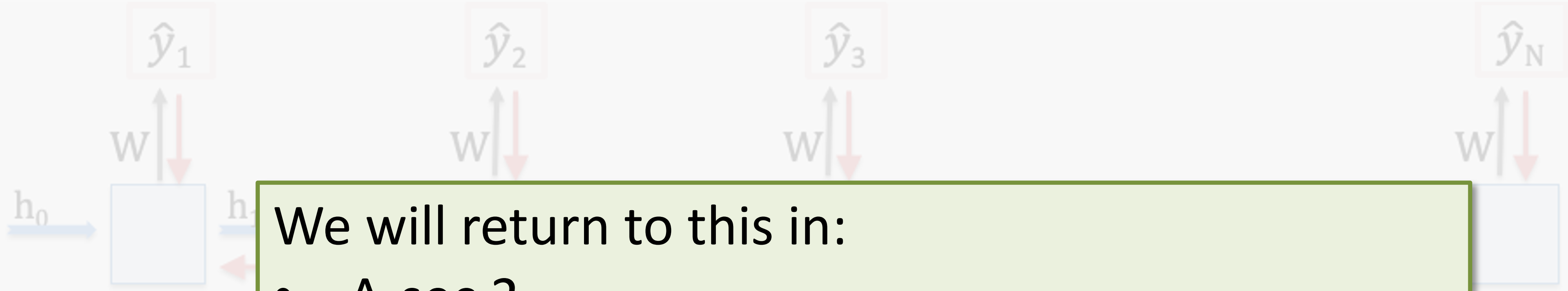
If many values < 1, then the product, i.e., the gradient, will be close to zero. This is called the **vanishing gradient problem**.

This causes the parameters to update very slowly.

If many values > 1, then the product, i.e., the gradient, will explode. This is called the **exploding gradient problem**.

This causes an overflow problem.

Training in RNN: **Backpropagation issues**



For longer senten

This requires the ollowing issues.

We will return to this in:

- A-sec 2
- GRU
- LSTMs
- Attention

For now, we will just acknowledge there is a potential problem with long sequences.

If many values < 1, t
gradient, will be clo
the **vanishing gradient problem.**

This causes the parameters to update very slowly.

roduct, i.e., the
called
the **exploding gradient problem.**

This causes an overflow problem.

Outline

Motivation behind RNNs

Introduction to RNN

Training in RNN

Bidirectional RNNs

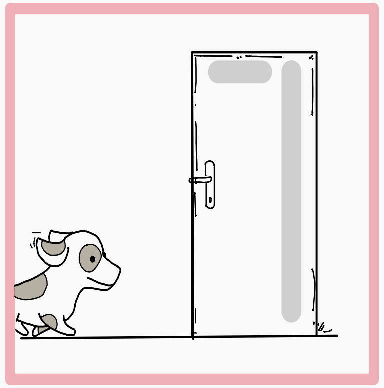
Deep RNNs

Flavors of RNN

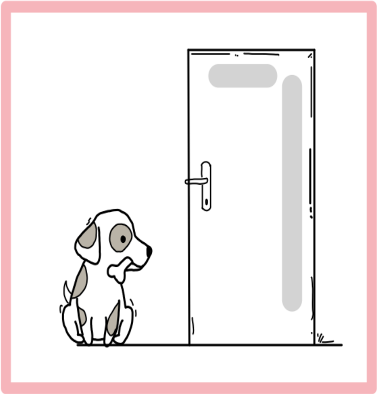
Bidirectional RNNs : **Motivation**

Given only Frame t-1 and t-2, it is difficult to predict the next frame t.

Frame t-2



Frame t-1

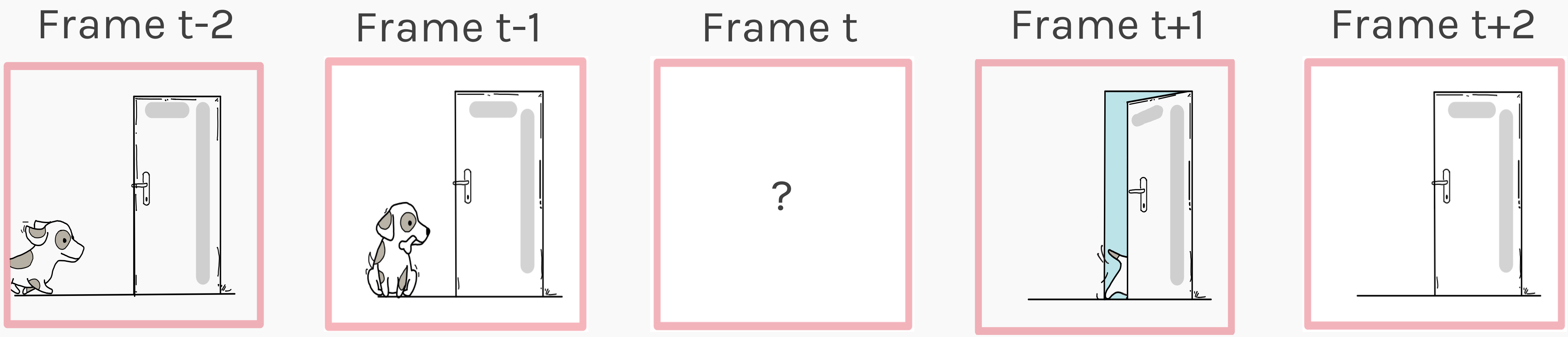


Frame t

?

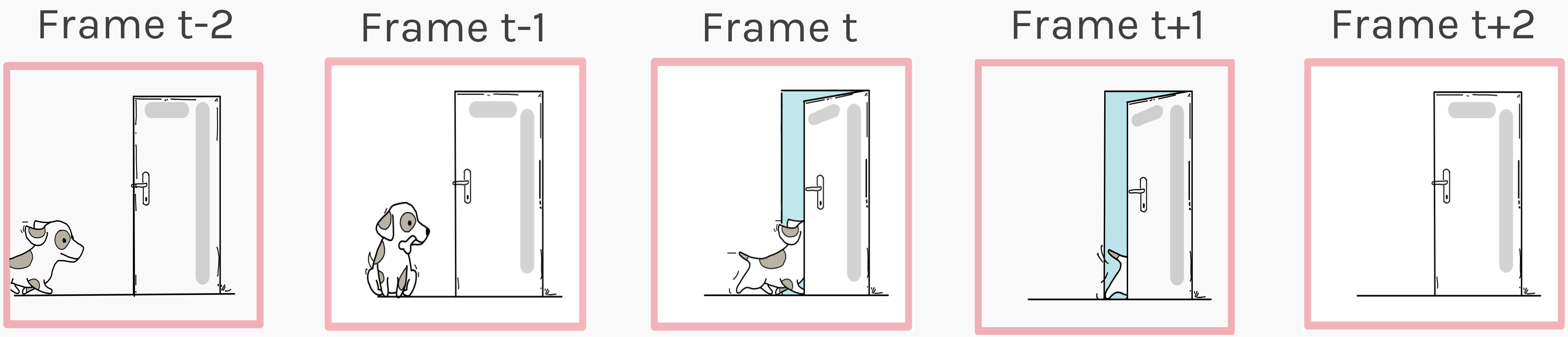
Bidirectional RNNs : **Motivation**

However, if we are to give some future frames it would be easier for the model to predict.



Bidirectional RNNs : **Motivation**

Thus, sequences after the one to be predicted play an important role to provide context for prediction.



However, simple RNNs process sequences only from left to right. They cannot look ahead into future sequences.

Bidirectional RNNs solve this problem by processing the sequence in both directions.

Bidirectional RNNs : **Motivation**

This is very important when we use RNNs for language modeling:

Consider the following:

*Pavlos said **he** needs a vacation.*

"he" here means *Pavlos* and we know this because *Pavlos* was before the word "***he***".

However, consider the following sentence:

***He** needs to work more, Mark said about Pavlos.*

We could not know the meaning of "***he***".

Bidirectional RNNs solve this problem by processing the sequence in both directions.

Bidirectional RNNs : **Motivation**

This is very important when we use RNNs for language modeling:

Consider the following:

*Pavlos said **he** needs a vacation.*

"he" here means *Pavlos* and we know this because *Pavlos* was before the word ***"he"***.

However, consider the following sentence:

***He** needs to work more, Mark said about Pavlos.*

We could not know the meaning of "***he".***

More on this later during the language model lectures.

Bidirectional RNNs solve this problem by processing the sequence in both directions.

Bidirectional RNNs

In a Bidirectional RNN there are two separate RNNs used: one for forward direction and one for reverse direction.

This results in a hidden state from each RNN, which are concatenated to form a single hidden state.

Hidden state of Forward RNN

Hidden state of Reverse RNN

$$h_t^F = \tanh(\ X_t\, V^F + h_{t-1}^F U^F +\ \beta_1^F)$$

$$h_t^R = \tanh(\ X_t\, V^R + h_{t+1}^R U^R +\ \beta_1^R)$$

$$h_t^T = h_t^F + h_t^R$$

Final hidden state

Concatenation not addition

The output equation is similar to that of simple RNNs:

$$\hat{y}_t\ = \sigma(h_t^T W +\ \beta_2)$$

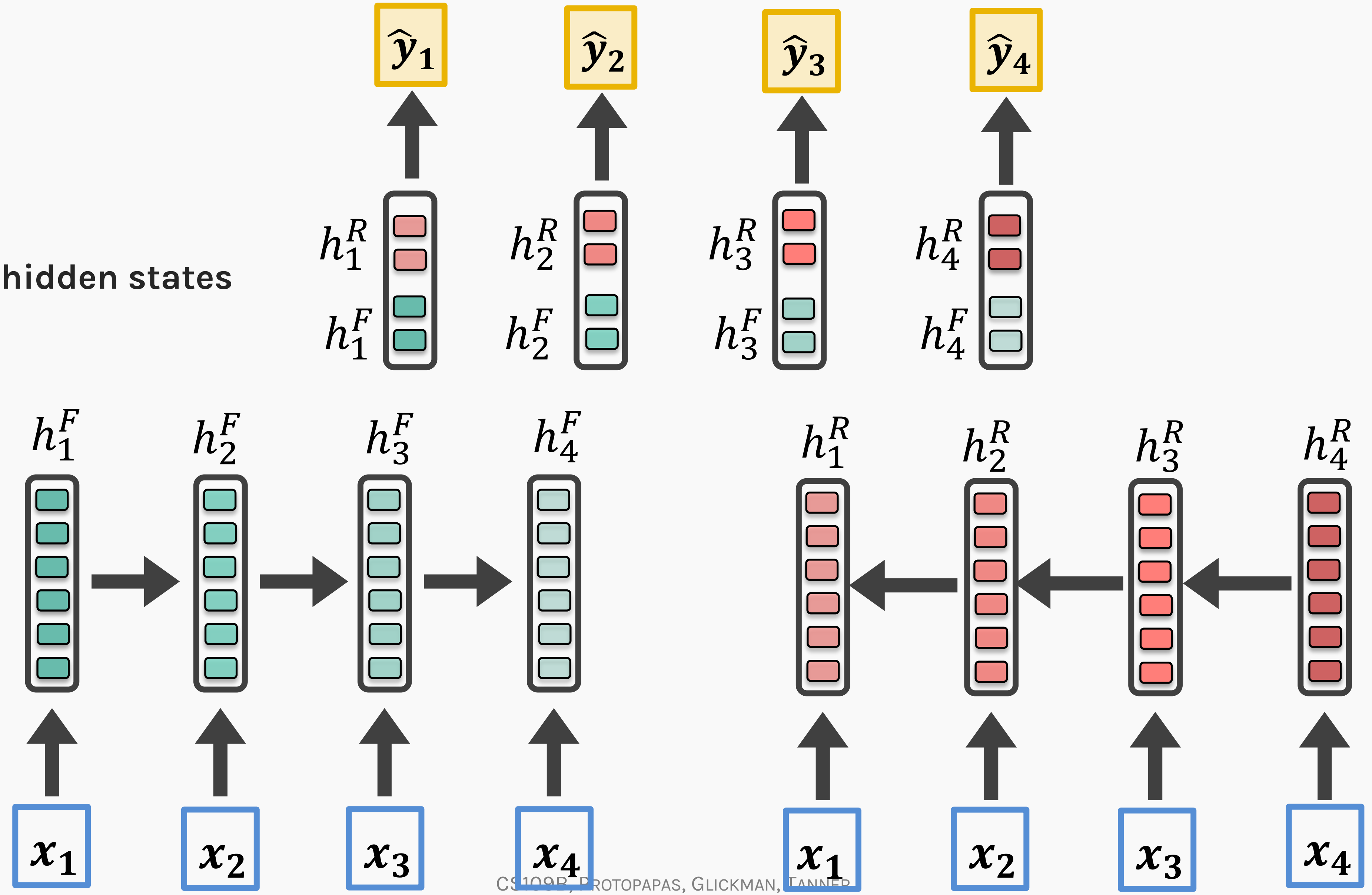
Bidirectional RNNs

Output layer

Concatenate the hidden states

Hidden layer

Input layer



Outline

Motivation behind RNNs

Introduction to RNN

Training in RNN

Bidirectional RNNs

Deep RNNs

Flavors of RNN

Deep RNN

RNNs units can be arranged in layers, so that the output of each unit is the input to the other units. This is called **a deep RNN**, where the adjective “deep” refers to these multiple layers.

Deep RNN

RNNs units can be arranged in layers, so that the output of each unit is the input to the other units. This is called **a deep RNN**, where the adjective “deep” refers to these multiple layers.

- Each layer feeds the RNN on the next layer

Deep RNN

RNNs units can be arranged in layers, so that the output of each unit is the input to the other units. This is called **a deep RNN**, where the adjective “deep” refers to these multiple layers.

- Each layer feeds the RNN on the next layer
- First time step of a feature is fed to the first layer RNN, which processes that data and produces an output (and a new state for itself).

Deep RNN

RNNs units can be arranged in layers, so that the output of each unit is the input to the other units. This is called **a deep RNN**, where the adjective “deep” refers to these multiple layers.

- Each layer feeds the RNN on the next layer
- First time step of a feature is fed to the first layer RNN, which processes that data and produces an output (and a new state for itself).
- That output of the first layer is fed to the next RNN, which does the same thing, and the next, and so on.

Deep RNN

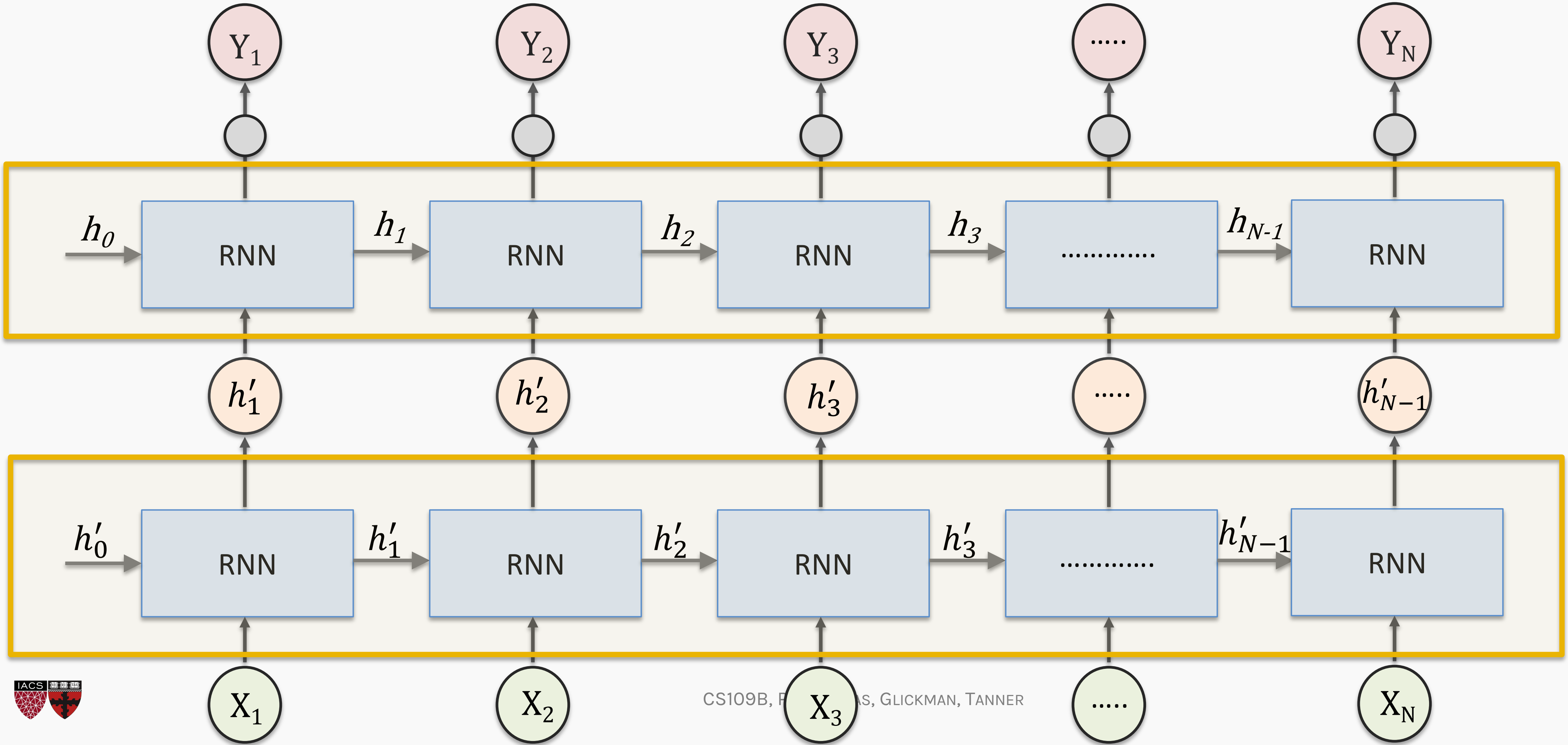
RNNs units can be arranged in layers, so that the output of each unit is the input to the other units. This is called **a deep RNN**, where the adjective "deep" refers to these multiple layers.

- Each layer feeds the RNN on the next layer
- First time step of a feature is fed to the first layer RNN, which processes that data and produces an output (and a new state for itself).
- That output of the first layer is fed to the next RNN, which does the same thing, and the next, and so on.
- Then the second time step arrives at the first RNN, and the process repeats.

Deep RNN

Hidden layers provide an abstraction (holds "meaning").

Stacking hidden layers provides increased abstractions.



Outline

Motivation behind RNNs

Introduction to RNN

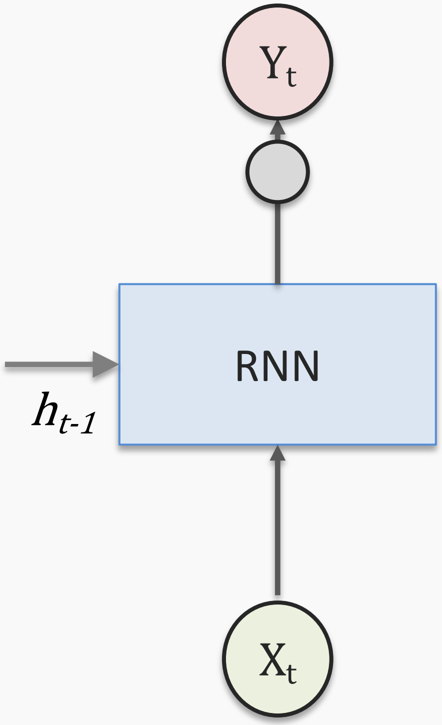
Training in RNN

Bidirectional RNNs

Deep RNNs

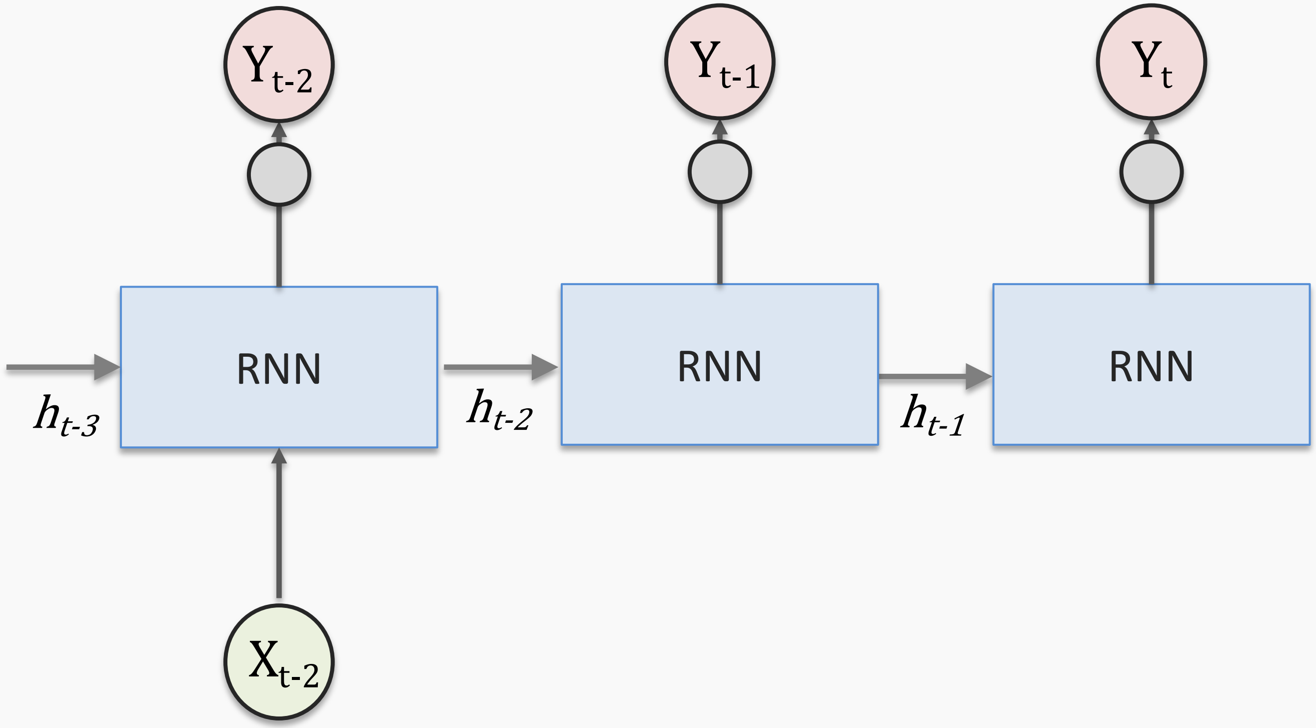
Flavors of RNN

Flavors of RNN : **One to One**



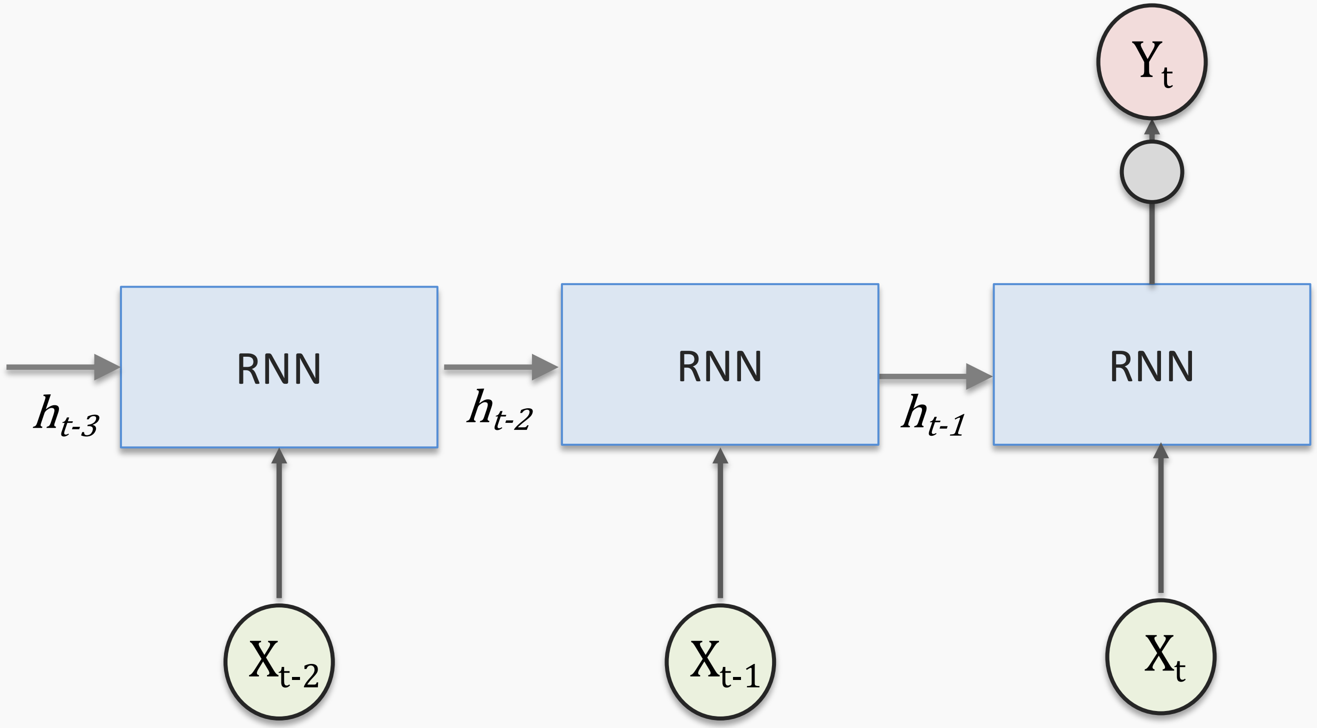
- The **One to One** structure is useless.
- It takes a single input and it produces a single output.
- Not useful because the RNN cell is making little use of its unique ability to remember things about its input sequence.

Flavors of RNN : **One to Many**



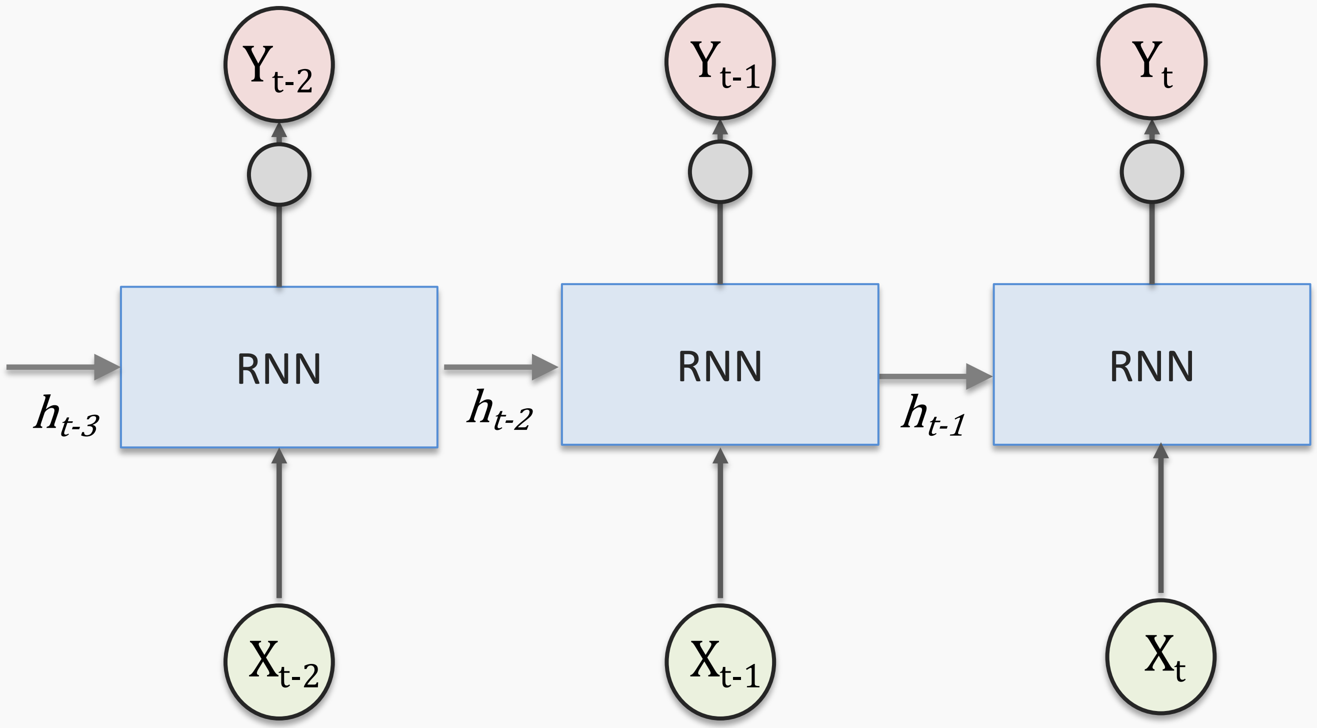
- The **One to Many** takes in a single piece of data and produces a sequence.
- For example, we give it the starting note for a song, and the network produces the rest of the melody for us.

Flavors of RNN : **Many to One**



- The **Many to One** structure reads in a sequence and gives us back a single value.
- Example: **Sentiment analysis**, where the network is given a piece of text and then reports on some quality inherent in the writing. A common example is to look at a movie review and determine if it was positive or negative.
- This structure is also used for any kind of **classification**.

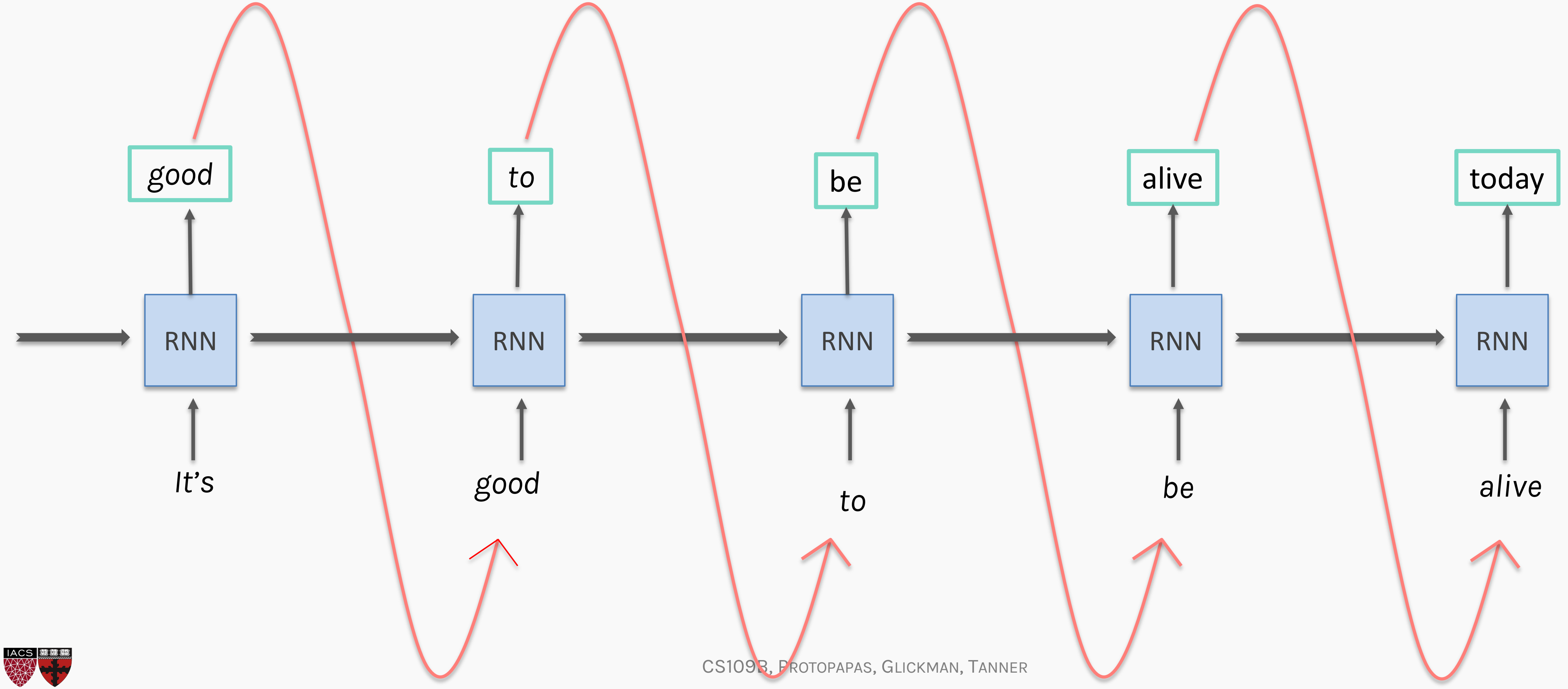
Flavors of RNN : **Many to Many**



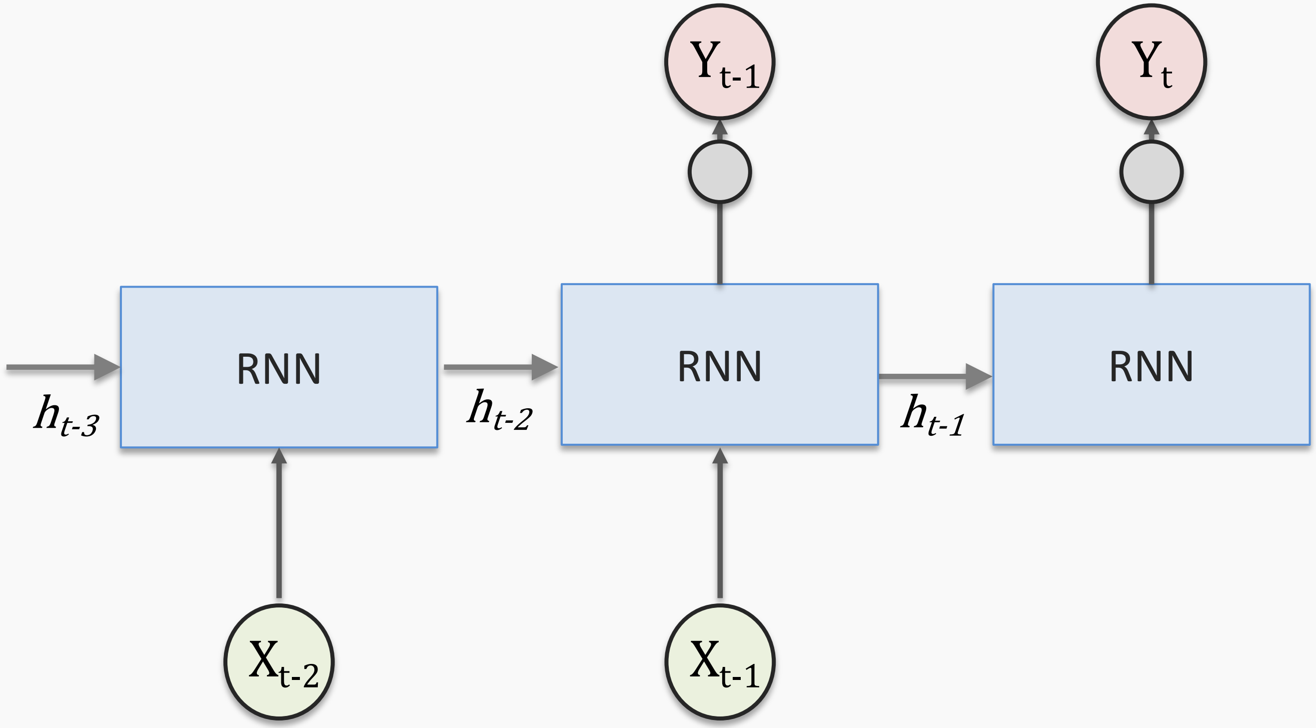
- The **Many to Many** structures are in some ways the most interesting.
- Examples:
  - Predict if it will rain given some inputs.
  - Language Model

Flavors of RNN : **Many to Many Inference**

Many to Many structures are used for language modeling where during inference the output of one unit is given as the input to the next unit.



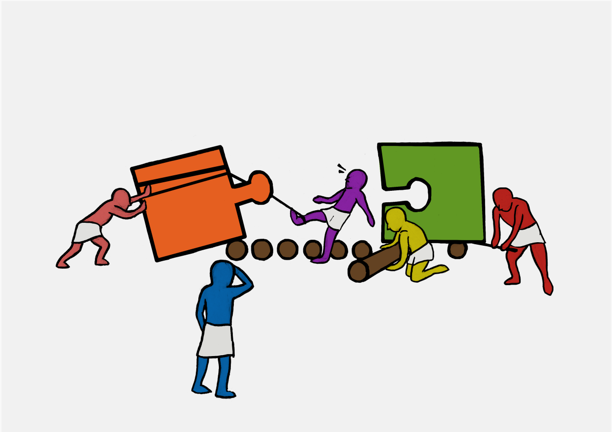
Flavors of RNN : **Many to Many**



- This form of **Many to Many** can be used for machine translation.
- For example, the English sentence: **"The white dog jumped over the cat"**

  In Spanish would be:
  "El perro blanco salto sobre el gato"
  In Spanish, the adjective "blanco" (white) follows the noun "perro" (dog), so we need to have a buffer so we can produce the words in their proper order in Spanish.

Exercise: RNN from scratch



The aim of this exercise is to understand what happens within an RNN unit that is wrapped within the tensorflow.keras.layers.SimpleRNN

The idea is to write a Recurrent Neural Network from scratch that generates names of dinosaurs by training on the existing names character-wise.